Red Hat Enterprise Linux 4 Application Compatibility

By Donald Fischer

Abstract

The Red Hat Enterprise Linux product family provides a powerful, stable platform for third-party software application developers. This document describes the different types of compatibility guarantees that are available to application developers building software for Red Hat Enterprise Linux on one or more hardware architectures. It also provides guidance to developers on best practices that should be followed in order to ensure application compatibility with future releases of Red Hat Enterprise Linux. Finally, it summarizes the compatibility policies followed by Red Hat within and between major releases of the Red Hat Enterprise Linux platform.

February 2005

Table of Contents

Types of Compatibility

There are multiple aspects of compatibility for Red Hat Enterprise Linux:

- *Application Compatibility:* indicates whether applications will compile and run across different instances of the operating environment, including updated versions and new releases, on a particular hardware architecture.
- *Configuration and Data File Compatibility:* which indicates whether configuration files and data files can be used among different releases of the operating environment.

Application Compatibility

Application compatibility specifies the portability of application source code and compiled application binaries across different instances of a computer operating environment.

Application compatibility can be broken down into two broad categories:

- *Source Application Compatibility*, which specifies whether application source code will compile and execute correctly across different instances of the operating environment. Source compatibility is defined by conformance with specified Application Programming Interfaces (APIs).
- *Binary Application Compatibility*, which specifies whether compiled application binary executables and Dynamic Shared Objects (DSOs) will run correctly across different instances of the operating environment. Binary compatibility is defined by conformance with specified Application Binary Interfaces (ABIs).

Red Hat's definition of source compatibility APIs and binary compatibility ABIs is such that there is a "contract" between the application and Red Hat Enterprise Linux operating environment. The difference between source compatibility APIs and binary compatibility ABIs is where the contract is enforced – at compile-time or runtime.

Source Application Compatibility

Source compatibility enables a body of application source code to be compiled and operate correctly on multiple instances of an operating environment, across one or more hardware architectures, as long as the source code is compiled individually for each specific hardware architecture. (Note that some platforms support more than one hardware architecture, as discussed later in section "Multiple Architecture Platforms").

Source compatibility is defined by an Application Programming Interface (API), which is a set of programming interfaces and data structures which are provided to application developers. For C-like languages, the programming syntax of APIs are defined in header files that specify data types and programmatic functions implemented by the operating system or libraries and made available to programmers for use in their applications. The syntax of APIs are enforced at compile time, or when the application source code is compiled to produce executable binary object code.

Source compatibility APIs are classified as:

- *De facto standards* - not formally specified but implied by a particular implementation
- *De jure standards* - formally specified in standards documentation

Red Hat Enterprise Linux provides backwards compatibility with de facto standards for core system components wherever possible. This means that an application built on one major release of Red Hat Enterprise Linux will continue to work throughout the product life cycle on subsequent updates of that major release and on the next major release of Red Hat Enterprise Linux as well. For example, applications that are are compiled with header files and linked to a particular version of glibc, the GNU C Library, are intended to continue to work with later versions of glibc. For the case of glibc, this is accomplished by providing versioned symbols, whose syntax and semantics are preserved in subsequent releases of the library even if a new, otherwise incompatible implementation is added. For other core system components, such as all 2.x releases of the GTK+ toolkit, backwards compatibility is ensured simply by limiting changes, which preserve the syntax and semantics of the defined APIs. In many cases, multiple versions of a particular library may be installed on a single system at the same time to support different versions of an API. An example is the inclusion of both Berkeley Database (db) version 4.2.52 and version 4.1.25 in Red Hat Enterprise Linux 4, each with its own set of headers and libraries.

In all cases, application developers should seek to ensure that any behavior they depend on is described in formal API documentation, so as to avoid introducing dependencies on unspecified implementation-specific semantics or even introducing dependencies on bugs in a particular implementation of an API. For example, new releases of the GNU C library are not guaranteed to be compatible with older releases if the old behavior violated a specification.

Red Hat Enterprise Linux by and large seeks to implement source compatibility with a variety of de jure industry standards developed for Unix operating environments. While Red Hat Enterprise Linux does not fully conform to all aspects of these standards, the standards documents do provide a defined set of interfaces, and many components of Red Hat Enterprise Linux track compliance with them (particularly glibc, the GNU C Library, and gcc, the GNU C/C++/Java/Fortran Compiler). There are and will be certain aspects of the standards which are not implemented as required on Linux.

A key set of standards that Red Hat seeks to conform with are those defined by the Austin Common Standards Revision Group (“The Austin Group”).

The Austin Group is a working group formed in 1998 with the aim of unifying earlier Unix standardization efforts including ISO/IEC 9945-1 and 9945-2, IEEE Standards 1003.1 and 1003.2 (POSIX), and The Open Group's Single Unix Specification. The goal of The Austin Group is to unify the POSIX, ISO, and SUS standards into a single set of consistent standards. The Austin Group includes members from The Open Group, ISO, IEEE, major Unix vendors, and the open source community.

The combined standards issued by The Austin Group carry both the IEEE POSIX designation and The Open Group's Technical Standard designation, and in the future the ISO/IEC designation.

More information on The Austin Group is available at http://www.opengroup.com/austin.

Binary Application Compatibility

Binary compatibility enables a single compiled application binary to operate correctly on multiple instances of an operating environment that share a common hardware architecture (whether that architecture support is implemented in native hardware or a virtualization layer).

Binary compatibility is defined by an Application Binary Interface (ABI). The ABI is a set of runtime conventions adhered to by all tools which deal with a compiled binary representation of a program. Examples of such tools include compilers, linkers, runtime libraries, and the operating system itself. The ABI includes not only the binary file formats, but also the semantics of library functions which are used by applications.

Similar to the case of source compatibility, binary compatibility ABIs can be classified into the following:

- *De facto standards,* which are not formally specified but implied by a particular implementation.
- *De jure standards,* which are formally specified in standards documentation.

Application Binary Interfaces specified by the GNU C, C++, Fortan and Java Compiler include the following:

- *Calling conventions*, which specify how arguments are passed to functions and how results are returned from functions.
- *Register usage conventions*, which specify how processor registers are allocated and used.
- *Object file formats*, which specify the representation of binary object code.
- *Size, layout, and alignment of data types*, which specifies how data is laid out in memory.
- *Interfaces provided by the runtime environment,* which must be available using the same name at all times and where the documented semantics do not change from one version to another

In addition, the Application Binary Interface for the GNU C++ Compiler specifies the binary interfaces for the following C++ language features:

- Name mangling
- Exception handling
- Invoking constructors and destructors
- Layout, alignment, and padding of classes
- Layout and alignment of virtual tables

The default system C compiler included with Red Hat Enterprise Linux 4 is derived from GCC 3.4 and is largely compatible with the C99 ABI standard. Deviations from the C99 standard in GCC 3.4 are tracked online at: http://gcc.gnu.org/gcc-3.4/c99status.html

The default system C++ compiler included with Red Hat Enterprise Linux 4 is derived from G++ 3.4 and conforms to a C++ ABI definition which is available online at: http://www.codesourcery.com/cxx-abi/

More information on the ABIs implemented by the standard Red Hat Enterprise Linux C and C++ compilers is available in the manual “Red Hat Enterprise Linux: Using the GNU Compiler Collection (GCC),” which is available online at:

http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/gcc/compatibility.html

Hardware Architecture Compatibility

Single Architecture Platforms

For applications that conform to specified interfaces, binary (ABI) compatibility is provided for all applications within a single hardware architecture. This means that application developers do not need to modify software to run on systems from different vendors, or with different hardware configurations, as long as they share a common hardware architecture and the software does not unconditionally use functionality of the architecture which is not universally available (such as special instructions which are available on only some processor models).

For example, applications compiled for Red Hat Enterprise Linux for the IBM iSeries and pSeries branded systems are supported on IBM POWER-branded systems with POWER5 processors because they share the same hardware architecture (ppc/ppc64).

Similarly, applications compiled for Red Hat Enterprise Linux for the AMD64 architecture are supported on Intel EM64T systems and vice versa because they share the same hardware architecture (x86-64).

Conversely, application binaries compiled for one hardware architecture typically will not run on other hardware architectures. For example, an application binary compiled for the Intel Itanium2 architecture will not run on an IBM POWER system. The exception to this rule is that Red Hat Enterprise Linux offers support for a compatibility runtime environment on some hardware architectures as discussed in the next section.

Red Hat Enterprise Linux supports a broad range of hardware platforms from multiple vendors across several hardware architectures. Hardware platforms that have been certified to run on Red Hat Enterprise Linux are described on the Red Hat Hardware Compatibility List (HCL), available at http://hardware.redhat.com/hcl.

Multiple Architecture Platforms

Red Hat Enterprise Linux offers native support for a broad range of hardware architectures, including 32-bit and 64-bit architectures. For some of the supported platforms, both a native hardware architecture and a compatibility hardware architecture runtime environment are supported.

For each native architecture supported by Red Hat Enterprise Linux, Table 1 indicates:

- The *native userspace architecture*, or the hardware architecture for which the native system applications and libraries are compiled.
- The *compatibility userspace architecture*, if applicable, which includes a subset of system applications and libraries supplied to offer runtime compatibility with other architectures. Note that not all system libraries included in the native userspace are included in the compatibility userspace.
- The *kernel architecture*, which is the hardware architecture for which the kernel is compiled.

Table 1: Native Architecture Support

Architecture	Native Userspace	Compatibility Userspace	Kernel
32-bit x86	32-bit x86 (.i386.rpm)	N/A	32-bit x86
64-bit AMD64 and Intel EM64T	64-bit x86-64 (.x86_64.rpm)	32-bit x86 (.i386.rpm)	64-bit x86-64
64-bit Intel Itanium2	64-bit Itanium2 (.ia64.rpm)	32-bit x86 (.i386.rpm)	64-bit Itanium2
64-bit IBM POWER	32-bit POWER (.ppc.rpm)	64-bit POWER (.pp64.rpm)	64-bit POWER
31-bit IBM S/390 Mainframe	31-bit S/390 (.s390.rpm)	N/A	31-bit S/390
64-bit IBM zSeries Mainframe	64-bit zSeries (.s390x.rpm)	31-bit S/390 (.s390.rpm)	64-bit zSeries

For the the AMD64 and Intel EM64T hardware architectures, either the 32-bit native distribution or the 64-bit native distribution can typically be installed on a system. When the 64-bit native distribution is installed, 32-bit application binaries are supported through the compatibility userspace. When the 32-bit native distribution is installed, only 32-bit binaries are supported and not 64-bit binaries.

For the Intel Itanium2 hardware architecture, 32-bit x86 application support requires the use of a software package called the IA-32 Execution Layer (IA32-EL). The IA32-EL is supplied by Red Hat as an RPM package on the Red Hat Enterprise Linux 4 “Extras” CD and via Red Hat Network for the Itanium2 architecture only.

For the IBM POWER hardware architecture, the default userspace is compiled for the 32-bit POWER hardware architecture, and a compatibility userspace is provided for 64-bit native applications. The kernel for the IBM POWER

architecture is 64-bit native.

In all of the cases depicted Table 1, the compatibility userspace contains a subset of the system libraries and a subset of Linux distribution applications as compared to the native userspace. The goal of the compatibility userspace is to provide a binary-compatible application runtime environment for the specified architecture. The compatibility userspace does not provide a complete application development environment. For best results, Red Hat recommends that applications be developed and compiled in in a native hardware environment whenever possible, rather than in a compatibility environment.

Configuration and Data File Compatibility

The Red Hat Enterprise Linux distribution contains over a thousand individual software packages, which implement a variety of different system services and capabilities. The majority of these applications are developed by active open source developer communities, whose policies and practices differ substantially from one to the next. One area in which community projects tend to differ is their commitment to preserving configuration file and data file formats between releases.

Configuration Files

Many of the packages in the Red Hat Enterprise Linux distribution include the concept of a configuration file which specifies application settings, typically in an application-defined text or binary data format.

Wherever possible, Red Hat seeks to preserve the stability of configuration file formats within a major release of Red Hat Enterprise Linux. This means that for example, when updating a particular package from Red Hat Enterprise Linux 4 Update 1 to Red Hat Enterprise Linux 4 Update 2, configuration file customizations made by a user or system administrator should continue to function as intended without manual intervention.

The same guarantee can not be made for all packages for upgrades from one major release to another (for example, from Red Hat Enterprise Linux 3 to Red Hat Enterprise Linux 4). In the case of an automated upgrade between major releases, configuration files from the previous release will typically be preserved, but may require manual administrator or user adjustment to work correctly with the version of the package in the newer major release, if configuration file formats have changed for that package.

Application developers are advised in general not to depend on the format of configuration files used by system packages, unless the configuration files are defined by a specification and the corresponding upstream community project has expressed a commitment to supporting those configuration file formats in all future releases.

Data Files

Similar to the situation with configuration files, many packages in the Red Hat Enterprise Linux distribution depend on data files with a specific text or binary data representation.

Wherever possible, Red Hat seeks to preserve the stability of data file formats within a major release of Red Hat Enterprise Linux. This means that for example, when updating a particular package from Red Hat Enterprise Linux 4 Update 1 to Red Hat Enterprise Linux 4 Update 2, data files created by earlier revisions of packages should continue to function as expected.

The same guarantee can not be made for upgrades from one major release to another (for example, from Red Hat Enterprise Linux 3 to Red Hat Enterprise Linux 4). The support for data file formats between major releases is typically dependent on the development policies of the upstream open source community project. For some packages, data file formats will always be maintained. For other packages, a compatibility mode is available for inter-operating with older data file formats. Finally, some packages will automatically upgrade data files from an old format to a new format.

Some specific examples for upgrades from Red Hat Enterprise Linux 3 to Red Hat Enterprise Linux 4 are as follows:

- The Evolution mail and groupware client and the Mozilla Firefox web browser will automatically upgrade user configuration and data files from earlier releases to the most recent format, the first time they are run for each user.
- The OpenOffice.org office suite includes support for reading and writing data files created by earlier releases of the office suite.
- The format of data files used by the kernel system call auditing infrastructure has changed between Red Hat Enterprise Linux 3 and Red Hat Enterprise Linux 4 due to the migration to a new auditing implementation in the kernel. Thus, audit data files and analysis tools from Red Hat Enterprise Linux 3 are not automatically inter-operable with those from Red Hat Enterprise Linux 4.
- The format of files created by the kernel in the /proc filesystem is not guaranteed across releases, unless otherwise specified by the kernel documentation. The location and format of individual files in the /proc filesystem may have changed between Red Hat Enterprise Linux 3 and Red Hat Enterprise Linux 4.
- Database content created with earlier releases of PostgreSQL can be migrated to the newer release by dumping the contents of the database to a text format and then importing the text files into a newer release.

Application developers are advised in general not to depend on the format of data files used by system packages, unless the data files formats are defined by a specification and the corresponding upstream community project has expressed a commitment to supporting those data file formats in all future releases.

If an application uses a system library to administer data files (such as databases), the application should be able to recognize changes in the data format as long as the system library interface hasn't changed.

Designing Software for Compatibility

Use of System Libraries

The Red Hat Enterprise Linux distribution contains over a thousand individual software packages, many of which include libraries that are available to developers for static or dynamic linking into applications.

Red Hat recommends that application developers avoid static linking whenever possible. Some of the disadvantages of static linking include the following:

- Bug fixes and security fixes must be applied multiple places if code is duplicated.
- Security measures such as load address randomization are not available to statically linked libraries.
- Statically linked applications are less efficient users of physical memory, since common dynamically linked code can not be shared among multiple application images.
- Some library functionality and developer tools are not applicable to statically linked applications.

Core Libraries

Red Hat defines a set of libraries whose APIs and ABIs will be preserved for each architecture across major releases of the distribution. To ensure application runtime compatibility across major releases (for example between Red Hat Enterprise Linux 3 and Red Hat Enterprise Linux 4), application developers are encouraged to limit their applications to linking against this limited set of libraries.

For Red Hat Enterprise Linux, the core set of libraries includes:

- libc, libgcc, libstdc++, libdl, libm, libutil, libcrypt, libz, libpthread, libncurses
- libX11, libXext, libXt, libICE, libSM, libGL
- libgtk, libgdk, libgdk_pixmap, libpango , libatk , libglib, libgmodule, libgthread, libgnomeprint, libgnomeprintui, libgconf, libglade

If an application can not limit itself to the interfaces of these core libraries, then to ensure compatibility across major releases, the application should bundle the additional required libraries as part of the application itself. In that case, the bundled libraries must themselves use only the interfaces provided by the core libraries.

Non-Core Libraries

Red Hat Enterprise Linux also includes a wide range of libraries whose APIs and ABIs are not guaranteed to be preserved between major releases. Compatibility of these libraries is, however, provided within a major release of the distribution. Applications are free to use these non-core libraries, but to ensure compatibility across major releases, application vendors should provide their own copies of these non-core libraries, which in turn should depend only on the core libraries listed in the previous section.

Packaging

For the best integration with the Red Hat Enterprise Linux distribution and software management tools, application developers are encouraged to package their software using the RPM Package Manager (RPM). RPM provides a robust software packaging mechanism that includes rigorous specification of application dependencies.

For improved compatibility across releases, application developers should follow these guidelines when creating RPM packages:

- Avoid using RPM triggers whenever possible.
- Don't depend on the execution order of preinstall or preuninstall scripts, which may change between releases.
- Explicitly state all required runtime and build dependencies using the appropriate RPM syntax.
- Do not modify, replace, or recompile files managed by Red Hat provided RPM packages.
- When considering dependencies, don't assume that all possible packages will be installed on every Enterprise Linux system. The default installed packages may change between releases.

The File Hierarchy Standard

Applications should follow the Filesystem Hierarchy Standard (FHS) when installing files. Specifically, third party software should install to a subdirectory of /opt. More information on the File Hierarchy Standard is available at: http://www.pathname.com/fhs/2.2/

Security-Enhanced Linux

Security-Enhanced Linux (SELinux) is a new capability in Red Hat Enterprise Linux 4. SELinux provides a Mandatory Access Control (MAC) system for Linux, which can be used to control access to system resources at a fine-grained level.

Red Hat Enterprise Linux 4 includes an SELinux policy known as the *targeted policy* that runs only a specific set of system services under SELinux protection. The targeted policy is designed so that it does not impact the runtime behavior of third-party applications. Application developers may also wish to investigate the development of SELinux policies for their applications to ensure additional enforcement of security at application runtime.

More information on the SELinux implementation in Red Hat Enterprise Linux 4 is available at: http://fedora.redhat.com/projects/selinux/

Red Hat Enterprise Linux Compatibility Policies

Compatibility Within A Major Release

One of the core goals of the Red Hat Enterprise Linux family of products is to provide a stable, consistent runtime environment for third-party applications. To support this goal, Red Hat seeks to preserve application binary compatibility, configuration file compatibility, and data file compatibility for all

package updates issued within a major release.

For example, a package update from Red Hat Enterprise Linux 4 Update 1 to Red Hat Enterprise Linux Update 2, or a package update that fixes an identified security vulnerability, should not break the functionality of deployed applications as long as they adhere to standard Application Binary Interfaces (ABIs) as previously discussed.

Compatibility Between Major Releases

Red Hat Enterprise Linux also provides a level of compatibility across major releases, although it is less comprehensive than that provided within a major release. With the qualifications given below, Red Hat Enterprise Linux 4 provides runtime compatibility support for applications built for Red Hat Enterprise Linux 2.1 and Red Hat Enterprise Linux 3.

Between major releases (for example, between Red Hat Enterprise Linux 3 and Red Hat Enterprise Linux 4), Red Hat seeks to provide binary application compatibility for applications that adhere to published standard ABIs and APIs referenced in earlier sections of this document. This statement applies both to native architecture support and compatibility architecture support.

Red Hat provides compatibility libraries for a set of core libraries. However, Red Hat does not guarantee compatibility across major releases of the distribution for dynamically linked libraries outside of the core library set unless versions of the Dynamic Shared Objects (DSOs) the application expects are provided (either as part of the application package or separate downloads). To ensure compatibility across major releases, application developers are encouraged to limit their dynamically linked library dependencies to those in the core library set, or to provide an independent version of the required non-core libraries packaged with their application (which in turn depend only on core libraries). As a rule, Red Hat recommends against statically linking libraries into applications.

Red Hat also reserves the right to remove particular packages between major releases. Red Hat provides a list of deprecated packages that may be removed in future versions of the product in the Release Notes for each major release. Application developers are advised to avoid using libraries on the deprecated list. Red Hat reserves the right to replace specific package implementations in future major releases with alternative packages that implement similar functionality.

Red Hat does not guarantee compatibility of configuration file formats or data file formats between major releases of the distribution, although individual software packages may in fact provide file migration or compatibility support.

For more information, visit www.redhat.com or contact us at 1-888-REDHAT1 (US and Canada) / +1-919-754-3700 (international).