HP-UX compatibility

Technical white paper

Introduction

Since the introduction of the HP-UX operating system on PA-RISC in 1986, HP has maintained a strong commitment to delivering source and binary compatibility with each update of the operating system. This commitment is based on a tradition of delivering compatibility that dates back to the introduction of the MPE operating system and the HP 3000 product line in 1974.

As applications have grown increasingly more complex and more dependent on third-party libraries, the concepts of source and binary compatibility have also grown more complex. Software developers and end users now require more than the simple assurance that a new OS release provides source and binary compatibility with the previous one: they need to be able to mix libraries from different third parties that may have been built with differing compilers and on different OS releases. With the introduction of Itanium®-based systems, they also require a new set of compatibility guarantees: binary compatibility across architectures, and compatibility of data structures as stored on disk or transmitted across networks. With these new needs, HP has formalized the concepts of compatibility and put in place processes to ensure that each release of HP-UX meets those needs.

This white paper describes these types of compatibility, HP's commitment to compatibility, and the actions that have been taken to deliver on that commitment for HP-UX 11i v2.

Types of compatibility

Any promise or expectation of compatibility is made for an application or a class of applications with respect to a change in the system upon which that application is built or deployed. The event that causes the change may be a new major or minor release of HP-UX, a software patch, a move from one hardware platform to another, or even a move from one programming environment to another (e.g., 32-bit to 64-bit). These events can affect compatibility in various ways.

Source compatibility

Source compatibility (for a given class of applications with respect to a given event) means that an application can be recompiled, in its entirety and without change (to its source code or its Makefiles), and the resulting binary will operate exactly as before. Preserving source compatibility requires careful attention to compilers, system header files, system libraries (archive or shared), system commands, system data files, and the kernel.

Binary compatibility

Binary compatibility means that an application, consisting of an executable and any application shared libraries, will execute exactly as before, without recompilation or other change. Binary compatibility also applies to scripts written in any supported scripting language (e.g., shell, awk, make). Where a binary compatibility guarantee applies, an application can be developed in one environment and deployed in a different one. Preserving binary compatibility requires careful attention to system shared libraries, system commands, system data files, and the kernel.

Data compatibility

Data compatibility means that an application built and deployed in one environment can exchange data (through disk files or across a network) with a corresponding application built and deployed in another environment. With the transition from PA-RISC to Itanium-based systems, this is a particularly important concern. Preserving data compatibility requires careful attention to compilers and system header files.

Relocatable object compatibility

Relocatable object compatibility means that an application can be recompiled in part, and relinked with a mixture of relocatable object files from differing build environments. Preserving relocatable object compatibility requires careful attention to compilers, object file formats, system header files, system libraries (archive or shared), system commands, system data files, and the kernel. When relocatable object compatibility is absent, all software must be recompiled, and ISV applications that sit on top of a stack of other ISV software must wait until all their dependencies have recompiled. This situation is called a "class move," and results in a long adoption process for the new environment.

Forward vs. backward compatibility

Normally, when we speak of compatibility, we are talking about a move in the forward direction—for example, a move from one release of HP-UX to the next, or from one processor architecture to a newer one. For patch releases (see below), we also talk about compatibility in the reverse direction: backward compatibility means that an application can be developed in one environment (e.g., on a system containing a certain patch), and deployed in an earlier environment (e.g., on a system not containing the patch).

HP’s commitment

HP supports different levels of compatibility for different kinds of change. The compatibility guarantees for each type of OS release and for other transitions are summarized below.

Major release

A major release of HP-UX adds major new functionality, and usually corresponds to a “.0” release (e.g., 11.0). Major releases are expected to occur infrequently—no more often than every 18 months or so—and have the most relaxed compatibility requirements for applications moving from the previous release. A major release requires binary compatibility and data compatibility for applications running on the previous release. Source compatibility, while important, may be compromised in order to provide new features or to conform to new standards. Relocatable object compatibility is not guaranteed, meaning that major releases imply a class move.

Minor release

A minor release of HP-UX adds new functionality, but nothing significant enough to justify a class move.   A minor release requires source compatibility, binary compatibility, data compatibility, and relocatable object compatibility for applications running on the previous release.

Patch release

A patch release of HP-UX usually is issued to fix defects, but can also be used to add new functionality in a limited manner. A patch release has the compatibility requirements of a minor release plus one of backward compatibility for relocatable object files within the patch stream (i.e., back to the minor release to which the patch has been applied). In other words, a relocatable object file compiled in a patched environment must be fully compatible with relocatable object files compiled in the corresponding unpatched environment. Furthermore, the resulting executable must be compatible with the unpatched environment. An exception to this requirement is made if the source code or makefiles are changed to take specific advantage of a new feature introduced with the patch. Thus, an application modified and recompiled to take advantage of a new feature is necessarily restricted from executing in environments where the patch supporting that feature has not been applied.

Migration to Itanium-based systems

The rapid migration of applications from PA-RISC to Itanium-based systems is dependent upon source compatibility, binary compatibility, and data compatibility. This migration requires these forms of compatibility between corresponding releases of HP-UX on the two architectures. Binary compatibility ensures that a PA-RISC application that runs on a given release of HP-UX will run unchanged on the corresponding release of HP-UX on an Itanium-based system (and, by extension, subsequent releases). Source compatibility ensures that an application can be recompiled without change to take advantage of full native-mode performance. Data compatibility ensures that a customer can deploy a mixture of PA-RISC applications and Itanium-based applications that share data without the need for any custom file translation utilities.

Migration to a 64-bit programming environment

The migration from a 32-bit to a 64-bit programming environment is usually a major undertaking for any application of significant size. Source compatibility and data compatibility cannot be guaranteed, since fundamental data types are changed in size. Beyond the scope of the changes in data model, however, HP does ensure a high degree of source compatibility to minimize developer effort during the migration. Binary compatibility applies only in the sense that all new 64-bit HP-UX platforms continue to support 32-bit applications. Relocatable object compatibility between the two environments is not supported.

The compatibility promises that HP makes to its customers are limited to “well-behaved” applications. A well-behaved application:

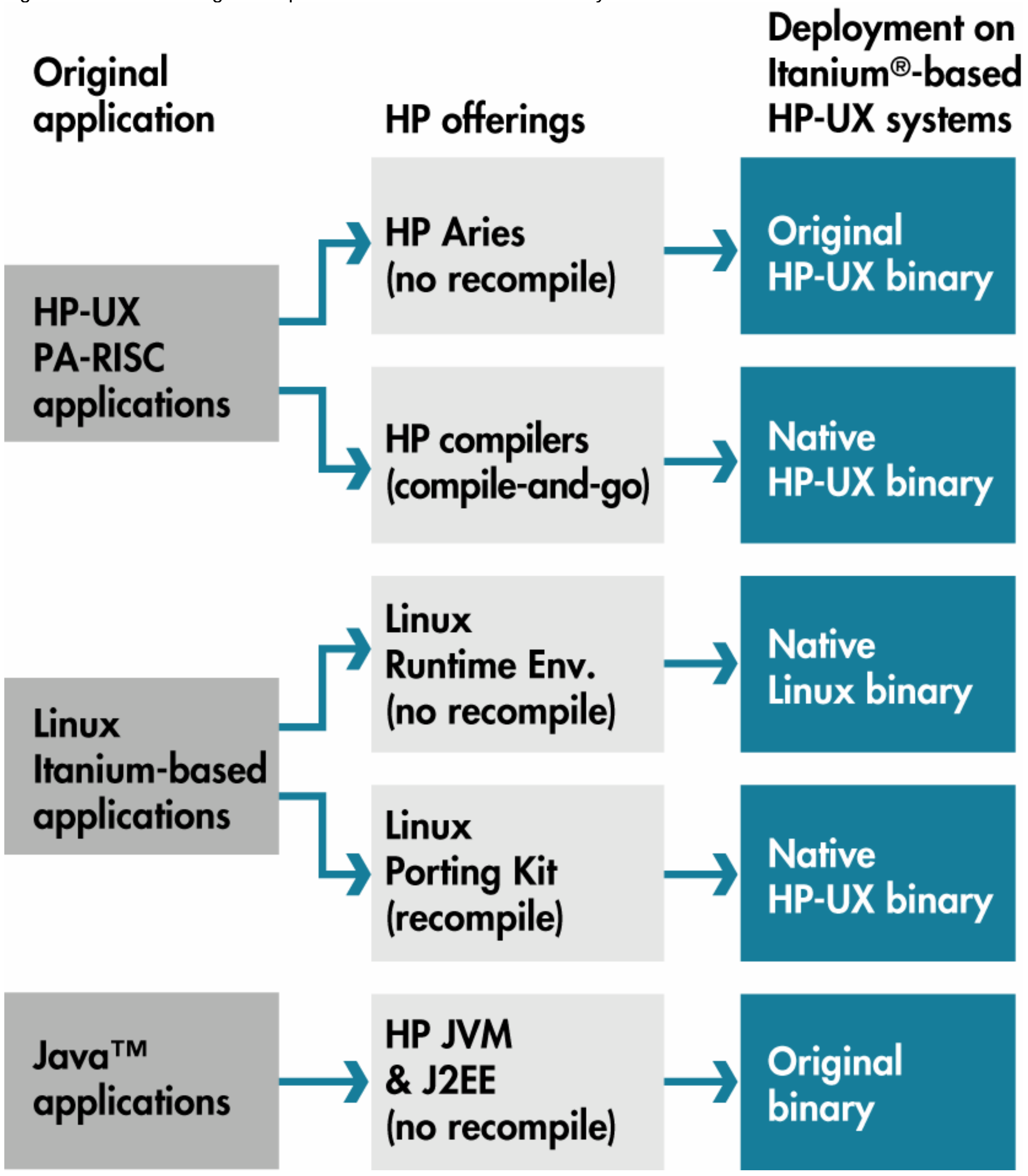
- must not use undocumented or private system APIs;
- must not rely on undocumented features;
- must not rely on behavior documented as platform-, architecture-, or configuration-specific;

- must not use part of an HP-UX product without using the whole product (e.g., extracting members of an archive library), except as officially supported by HP (e.g., the HP-UX linker).

Migrating from PA-RISC to Itanium-based systems

HP supports binary, source, and data compatibility for PA-RISC applications making the transition to Itanium-based systems. For both binaries and sources, HP-UX Release 11.0 or later is the recommended transition point; an application or library that runs on 11.0 on PA-RISC can be migrated easily. Figure 1 illustrates the migration options available.

Figure 1: Seamless migration paths to Itanium-based HP-UX systems


Original application
HP offerings
Deployment on Itanium®-based HP-UX systems
HP-UX PA-RISC applications
HP Aries (no recompile)
Original HP-UX binary
HP compilers (compile-and-go)
Native HP-UX binary
Linux Itanium-based applications
Linux Runtime Env. (no recompile)
Native Linux binary
Linux Porting Kit (recompile)
Native HP-UX binary
Java™ applications
HP JVM & J2EE (no recompile)
Original binary

Binary compatibility between architectures

Most well-behaved applications built for PA-RISC will run without modification on Itanium-based systems. These applications execute under a software-enabled compatibility mode subsystem called "Aries." The Aries subsystem consists of a dynamic translator that translates PA-RISC instruction sequences into the native instruction set, and a system interface layer that emulates the PA-RISC system ABI. Aries is invoked automatically by the HP-UX kernel whenever a PA-RISC application is executed, and is transparent to the user of that application.

The Aries subsystem delivers near-hardware levels of reliability, supports multi-threaded applications, and, since it runs at the same privilege level as the application, no additional security risks are introduced.

Some applications may be dependent on specific features of the hardware or the operating system, and may therefore be unsupported in compatibility mode. For a list of the limitations of the Aries subsystem, see the Appendix.

The Aries subsystem does not support applications that contain a mixture of PA-RISC code and native code within the same program ("mixed-mode" programs). Such applications must be partitioned manually into two separate programs, one that consists entirely of PA-RISC code, and another that consists entirely of native code. For a complete discussion of this technique, see the white paper "HP-UX 11 v2 and mixed-mode programs running on Integrity servers—why and how." More information is also available at www.hp.com/go/hpux.

Because binary compatibility is achieved through dynamic instruction translation, the performance of an application running in compatibility mode will be less than that of the same application recompiled and running in native mode. Applications that are largely interactive or I/O intensive should experience little to no noticeable degradation in performance, while those that perform heavy computation may run noticeably slower than on a recent PA-RISC system. We recommend recompilation for all applications and libraries where performance is a concern.

Applications compiled for Itanium-based systems cannot be executed on PA-RISC systems.

Source compatibility between architectures

HP has issued the following statement of source compatibility between the two hardware platforms:

> As of HP-UX 11i v1.6, ISV and customer applications* that are supported on HP-UX 11i on PA-RISC will compile and execute correctly on Itanium-based systems with no changes to the source code.
>
> *Applications must be well behaved and free of explicit dependencies on the PA-RISC architecture.

In order to meet this goal, HP has taken the following steps:

- The C++ compilers for the two architectures share common front-end source code.
- HP-UX header files and system APIs are based on shared source code.
- HP-UX supports 32-bit applications on Itanium-based systems, so applications do not need to port to 64-bit.
- All HP-UX compilers and libraries must undergo compatibility testing.
- Incompatibilities are subject to a thorough review process and are allowed only when necessary. Once allowed, they are documented as compatibility exceptions.
- The HP-UX operating system, its commands and libraries, and dozens of ISV applications, comprising tens of millions of lines of source code, have been compiled with the new compilers. Incompatibilities not already identified as exceptions have been treated as defects and fixed.

Of course, some applications may of necessity be non-portable, and may need some modification in order to make a successful migration to native execution. HP provides a Software Transition Kit (STK) on the developers portal Web page (http://www.hp.com/go/developers) that can be used to scan an application's source code to look for portability problems. For a list of known compatibility exceptions, see the Appendix.

Source and binary compatibility for Java applications

HP supports Java SDK/RTE Release 1.3.1.13 and Release 1.4.2.04 on both platforms. Java applications for these releases are fully compatible in both directions.

Migrating from Linux to HP-UX

Itanium-based HP-UX systems provide both source and binary compatibility for many Itanium-based Linux applications. Source compatibility is provided by the Linux Porting Kit, and binary compatibility is provided by the Linux Runtime Environment.

The Linux Porting Kit (LPK) provides tools and libraries to help port Linux applications to HP-UX with a minimum of source code changes (this product is available for both PA-RISC and Itanium-based HP-UX systems). For more details, see http://devrsrc1.external.hp.com/LPK/.

The Linux Runtime Environment (LRE) allows many Linux applications compiled for Itanium-based systems to execute on Itanium-based HP-UX systems without recompilation and without emulation. The LRE provides support for Linux system calls and signals on HP-UX, and supports debugging of Linux binaries. Support for Linux applications is subject to the following limitations:

- The application must be shared bound.
- The application must be compiled with gcc 2.96 or later.
- The application must be an Itanium-based 64-bit binary (the LRE does not support emulation of 32-bit x86 binaries).
- No support is yet provided for the /proc filesystem or the ptrace system call.
- Support is limited to user-space applications; device drivers are not supported.
- Applications that access persistent system files (e.g., /etc/utmp, /etc/wtmp) directly may not function, as the format of these files on HP-UX may differ from the format on Linux.

How HP delivers compatibility

Compatibility is considered a fundamental aspect of product quality. As part of the overall quality control, HP has put in place several processes to ensure the consistent delivery of compatibility with each operating system release. At the center of these processes is the Application Availability and Compatibility Team (AACT), which has responsibility for educating our internal developers about compatibility, reviewing changes for compatibility impact, and recommending implementation strategies for avoiding incompatible changes. Each development lab that contributes software to the HP-UX operating system names a representative to the AACT, and he or she acts as the compatibility champion for that lab.

The processes begin with compatibility training. The AACT has put together a comprehensive compatibility training package, which covers the importance of compatibility to our customers, the types of compatibility, typical mistakes that can lead to incompatibilities, techniques to prepare for future changes without incompatibility, and potential solutions to compatibility problems.

Compatibility concerns are addressed at the earliest stages of the product lifecycle, and each product is responsible for maintaining a compatibility statement from design to implementation to release. Each HP-UX system release has clearly-defined compatibility requirements, and each product must adhere to those requirements. When a compatibility problem (or a potential problem) is identified, it is sent to the AACT for review. The compatibility experts on the AACT may be able to identify a solution that avoids any incompatibility at all. Ultimately, the lab manager responsible for the product must approve any compatibility exception. Any exceptions are documented and placed in the release notes.

The AACT also maintains a dependency database that allows any development engineer to look up a system API to discover all known products (including third-party products) that depend on that API. This database is used to aid in obsolescence decisions and to help identify potential problems with proposed API changes.

Conclusion

HP's commitment to source and binary compatibility ensure a smooth and quick migration from PA-RISC to Itanium-based systems, and from each OS release to the next. The high level of source compatibility means that performance-sensitive applications can take advantage of the performance advantage of the Intel® Itanium architecture as quickly as possible, and the binary compatibility provided by Aries enables rapid customer transition to the new systems.

Table 1. HP-UX 11i releases

Release name	Release identifier	Supported processor architecture
HP-UX 11i v1	B.11.11	PA-RISC
HP-UX 11i v1.5	B.11.20	Intel Itanium
HP-UX 11i v1.6	B.11.22	Intel Itanium
HP-UX 11i v2	B.11.23	Intel Itanium and PA-RISC

Appendix: Limitations

The Aries compatibility mode environment is subject to the following limitations:

- Only pure PA-RISC applications are supported. No mixing of PA-RISC and native code is possible.
- Kernel- and architecture-dependent code is not supported. Any existing PA code that depends on internal kernel data structures (including /dev/kmem), the machine state, or privileged-mode instructions must be recoded.
- PA-RISC applications compiled prior to HP-UX Release 9.0 are not supported.
- Applications that depend on timing-dependent behavior (e.g., real-time response) may not operate correctly.
- Applications that reference the B (Taken Branch) bit in the PA-RISC Processor Status Word are not supported.
- Applications that use the ptrace, ttrace, or profil system calls are not supported. (Aries does, however, support debugging of PA-RISC applications using gdb.)
- All NaNs will behave as Quiet NaNs. Signaling NaNs are not supported.
- Because Aries consumes a small amount of the application's virtual memory space, an application that uses nearly all available memory may not operate correctly.
- Applications that depend on the difference between fork and vfork are not supported.

The following limitations apply to migrating source code from PA-RISC to the new architecture as of HP-UX 11i v2:

- K&R C is no longer supported.
- Convex parallelization pragmas and library functions are no longer supported. (OpenMP is supported on both platforms as a replacement.)
- Architecture-specific code, options, and pragmas must be modified for the new architecture. These include
  PA-RISC assembly code, inline assembly operations, options and pragmas used for tuning code for the PA-RISC architecture and runtime, and calls to any system APIs that are supported only on PA-RISC.
- The #pragma HP_ALIGN is no longer supported. (#pragma pack, which is common to Gnu and Sun compilers, should be used instead.)
- Floating-point operations may result in slightly different results (Itanium-based systems will usually give greater accuracy), and applications may observe differences in the treatment of NaNs, denorms, infinities, signed zeroes, exceptions, and flush-to-zero.
- The use of any third-party library is subject to the availability and support of that library on the Intel Itanium Processor Family and HP-UX 11i v2. Native code and compatibility-mode code cannot be mixed within a
  single program.

07/2004

5981-7108EN

invent