
Chapter 18. Porting Applications to Linux on the Mainframe

As a general rule, it is easy to port UNIX applications (including Linux applications on other hardware architectures) to Linux on the mainframe. Some software vendors were surprised to discover that their application ran after just recompiling their source.³⁶ Of course, it is not always that easy. This chapter gives guidance on how to decide if a port is feasible for a particular application.

We address questions such as:

- Why port an application to Linux on the mainframe?
- What is required for porting an application?
- How much effort will the port take?
- Where to get more information?

18.1 What you can gain by porting an application to Linux on the mainframe

As with any project, assessing the Return of Investment for porting an application must balance the expected advantages against the cost. There are two different cases where porting promises to be profitable:

- Porting the application is very simple and means a small investment that can easily be recovered.
- The expected advantages of having the application on Linux on the mainframe are significant enough to justify the required investment.

If your company uses its own proprietary applications in production, it might be able to capitalize on the quality and flexibility of the mainframe hardware. It can also take advantage of zSeries features like virtualization, HiperSockets, the cryptographic facilities, or the 64-bit architecture. It might also be able to use the mainframe's virtual communication methods to achieve a closer integration of the application in an integrated server environment. See Chapter 19, "Building Integrated Server Environments."

³⁶ For a sample of reports on porting experiences, visit <http://www.ibm.com/servers/enable/linux/quotes.html>.

There is a growing demand for Linux applications for the mainframe. If you are a software vendor, having a Linux-on-the-mainframe version of your application can be a decisive competitive advantage. A number of major software vendors already have ported some of their strategic software to Linux on the mainframe.

18.2 Before you decide to port

This section discusses some key considerations for porting applications to Linux on the mainframe. Before choosing an application as a candidate for porting, you need to be aware of the application's dependencies on other software, decide which mainframe architecture to target (31-bit or 64-bit), and determine whether you want to provide support for different distribution levels and distributions.

18.2.1 Third-party dependencies

Whether porting an application is feasible also depends on the context in which the application is to run. The application might depend on supporting software. A port would then imply also porting that software. If the required supporting software is a proprietary application or middleware of a third party, the port becomes dependent on the willingness of that third party to port its software. StoreCompany, for example, required DB2 Connect (see 19.3, “Connectors to back-end systems”) for its online sales project.

18.2.2 Distribution dependencies

Companies or software vendors might want to build a distribution-neutral application. The effort for ensuring support by different distributions depends on how strongly the code depends on specifics of a distribution and on how different the distributions are. The additional effort is likely to be restricted to tests for each distribution where the application is intended to run.

18.2.3 31-bit versus 64-bit

Linux on the mainframe applications can be for the 31-bit or for the 64-bit architecture. In keeping with the mainframe principles, zSeries architecture supports all software that ran on its 31-bit predecessor. Applications that are built for 31-bit run on both zSeries and S/390 machines.

If an application requires any new capabilities of the zSeries architecture (for example, 64-bit addressing), it can be built only for 64-bit distributions. It is possible to run 31-bit

applications on a 64-bit distribution. However, a 31-bit application cannot use 64-bit shared libraries. All libraries required by the application must be provided in 31-bit format.

18.2.4 Architecture-dependent code

If application code makes implicit or explicit assumptions about the underlying hardware or software platform, the code will probably have to be adapted before it can run on Linux on the mainframe. This section points to some sources where interfaces are documented.

How Linux on the mainframe differs from Linux on other platforms is described in the ELF Application Binary Interface Supplement available on:

<http://www10.software.ibm.com/developerworks/opensource/linux390/index.shtml>.

zSeries assembler code for Linux follows the syntax conventions of the GNU assembler but uses zSeries (S/390) machine code. For reference information on zSeries machine code, see *z/Architecture Principles of Operation*.

Because zSeries is a big endian system application code that processes byte-oriented data that originated on a little endian system may need to be adapted.

For details, refer to

http://www.ibm.com/servers/esdd/articles/linux_s390/

and to

<http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130115.pdf>.

The effort for handling architecture-dependent code depends on how the application is structured. A well-designed application might have such code in only a few specific modules. In this case, the code sections to be adapted are isolated and easily located.

The architectural differences between the mainframe and other platforms are documented. If it is required, the documentation can be found and accessed on the Internet.

18.3 What effort to expect

The effort for porting an application strongly depends on the language it is written in and the platform it has been written for. It can be very simple, but in the worst case, it can amount to a rewrite.

If an application is entirely written in Java, it will run unchanged on any platform where the required Java environment is available. For applications or modules written in high-level

languages that are not compiled at runtime (such as C and C++), a recompile to the zSeries architecture is required. Applications or modules with operating system or hardware dependencies need to be adapted to account for the different interfaces.

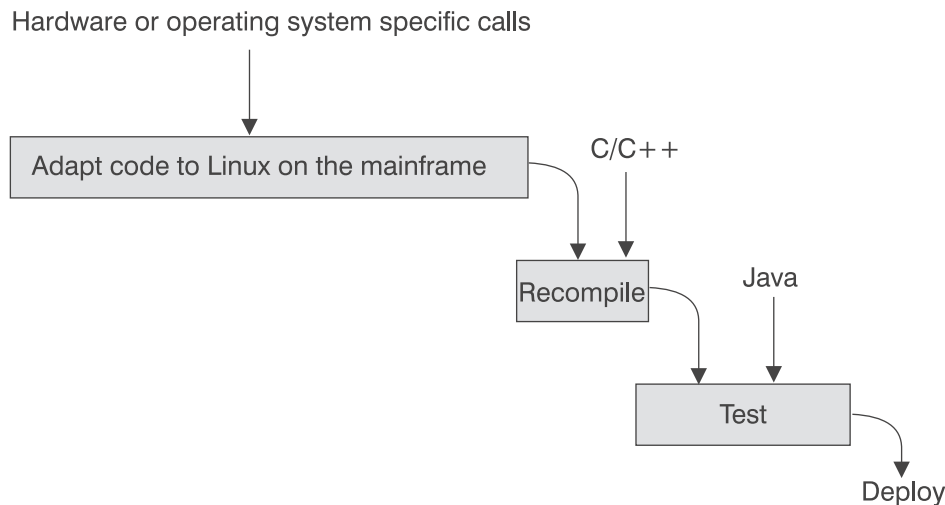


Figure 18-1. The effort for porting an application depends on the language in which it is written

In general, the lower the level of the program language and the more platform-specific features that are used by the code, the more porting effort is to be expected (Figure 18-1). In some cases, it can be advisable not to port but instead rewrite in a higher-level portable language, for example, C or Java.

18.4 What you need

This section summarizes which resources you will need for a porting project.

The skills required for porting an application to Linux on the mainframe are common software development skills. Your development team also needs the skills to use the Linux application development tools required to build your application. The standard Linux development tool chain (for example, compilers and editors) is included in the mainframe Linux distributions.

The GNU compiler and binary utilities can be configured to support a cross-compile environment. In cross-compiling, the target platform of the compilation is not the platform where the compilation runs. Application code can be compiled on a PC or low-end server but tar-

geted to the zSeries platform. Many companies do a significant portion of the porting effort on a workstation.

Naturally, the final test of the ported application needs to run in the target mainframe environment. Operating a zSeries test environment requires some zSeries and probably also z/VM skills.

If you already have a mainframe machine, an LPAR or z/VM guest can easily be set up as a test environment. If you do not have a mainframe, you might be able to get assistance from IBM. Visit IBM's Linux Community Development System Web site at <http://www.ibm.com/servers/eserver/zseries/os/linux/lcds/> to find out the terms and conditions for gaining access to IBM mainframes for testing your port.

18.5 Where to get more information

Visit the following site for technical documentation:

<http://www10.software.ibm.com/developerworks/opensource/linux390/index.shtml>.

Visit this site for links to useful tools: <http://www.ibm.com/developerworks/oss/linux/>.

For help, you can also contact one of IBM's porting centers, IBM Global Services, or a distributor.

18.6 Summary

Porting an application from UNIX to Linux can be a small effort, especially for Java or C/C++ applications. The ported application can take advantage of the mainframe hardware. For example, it can use HiperSockets or cryptographic facilities. As we will explore further in the next section, you can also use virtual networks to achieve a close integration with other applications on the same machine.

Most of the porting work can be done in your developers' favorite Linux environment. The final test requires access to the target Linux-on-the-mainframe environment. Given that sufficient resources are available, this means a z/VM test image that can easily be created.

The Linux distributions include much of what you need for porting an application. There are Web sites with tools, documentation, and other resources you might find helpful for porting. There is a possibility for gaining access to a Linux-on-the-mainframe test environment so you do not necessarily need your own mainframe machine to port an application to Linux on the mainframe.

