



Porting UNIX Applications to Linux - Hints and Tips

Table of Contents

Introduction	Page 2
Characteristics and values of Linux	Page 2
Key questions to consider before starting	Page 3
Will migration involve a huge initial investment?	Page 3
How much will it cost, and how long will it take?	Page 3
Will my application continue to work on the original UNIX platform?	Page 4
Porting roadmap for C and C++ applications	Page 4
Step 1: Downloads	Page 4
Step 2: Build your C/C++ application for Linux on Solaris	Page 5
Step 3: Build and test your application on Linux for Sun UltraSPARC	Page 5
Step 4 : Build and test your application on target Linux platform	Page 6
Porting roadmap for Java applications	Page 6
Step 1: Downloads	Page 7
Step 2: Use application on target Linux platform	Page 7
Run-time interfaces	Page 7
<i>System calls, C library and C++ library</i>	Page 7
<i>Desktop: CDE versus GNOME/KDE</i>	Page 8
<i>Threading and process management</i>	Page 8
<i>Operational considerations</i>	Page 8
Porting from Linux to Linux on zSeries	Page 9
Why porting to Linux on zSeries?	Page 9
Linux on zSeries porting hints and tips	Page 9
<i>Little endian to big endian</i>	Page 10
<i>Assembler code</i>	Page 10
<i>Absolute addresses and high-order bits</i>	Page 10
<i>Application development tools</i>	Page 10
<i>Miscellaneous</i>	Page 11
Summary	Page 11
Porting experiences - What others are saying	Page 12
For more information	Page 13

Introduction

Did you know that Linux® is the number two server operating system in the world today? Are your applications running on Linux today? If not, this paper is intended to give you hints and tips and describes a roadmap on how you can port applications to Linux. It is based on the experiences of software engineers from Independent Software Vendors (ISVs) and from the worldwide IBM® labs, who have been involved in many projects porting applications to Linux.

First we describe the characteristics of Linux and its specific value for developing and hosting applications. Next we review key questions and provide a porting roadmap that explains the major steps of a port. Finally we show how easy it can be to port from one Linux platform to another, using IBM ~ zSeries™ as an example. (The term zSeries refers to both: S/390® and zSeries.)

Characteristics and values of Linux

Openness means Application Flexibility: Linux is a genuinely open system, making application portability a reality. In many cases, applications can be brought to Linux with a simple recompile. This rapid portability gives you real flexibility in optimizing your development and deployment platforms.

Diverse Hardware Support means Platform Flexibility: Linux runs on nearly every known processor, whether RISC or CISC, 32-bit or 64-bit. It spans the entire range from the mainframe server (IBM ~ zSeries) to PowerPC®, Sun® UltraSPARC, Alpha, PA-RISC, Intel® processors, and on down to personal digital assistants and embedded systems. In particular, Linux is supported on all IBM ~ brand servers.

Source Code Availability means Do-it-yourself Flexibility: Unlike with most commercial software, which is distributed in binary form, Linux kernel source code is available. You can take the source code and modify and recompile it in order to meet your specific requirements. Debugging and resolving problems or adding new features is no longer dependent on any supplier. You can make the necessary changes yourself provided that they follow the provisions of the GNU General Public License.

Serious Design means Security: Linux security is built from the kernel up, and since the code that implements it is subject to the scrutiny of thousands of developers, weaknesses are discovered early. Linux comes with extensive routing and firewall capabilities built in.

Popularity means Existing Skill: Because Linux is popular in colleges and universities, most new graduates have Linux experience. Linux user groups in almost any city are a source of local contacts for skilled people, and Web sites and Internet news groups remain a valuable and effective source of Linux how-to information and informal technical consultation.

Key questions to consider before starting

Porting to Linux can be easy, simple and straight-forward, particularly if your UNIX® applications are written according to common open standards. If you think that a move to Linux is attractive, you need to analyze the potential costs and risks involved in the port and how to mitigate them.

Will migration involve a huge initial investment?

Costs - Will the porting involve a huge initial investment of time, people and money? Will the project freeze all other new work and consume entire teams? Are big capital outlay and retraining costs required up front? Is it an all-or-nothing proposition that once begun can only either be completed or backed out in full?

Porting to Linux is manageable - The Porting roadmap section of this paper outlines how porting can be staged one step at a time, where each intermediate step is stable by itself. This gives you the freedom to manage costs, people and projects according to your priorities. It means you can commit each step independently and reassess your priorities and goals after each one. The result is a much more manageable risk and less impact on your business.

How much will it cost, and how long will it take?

Costs in time and money - Thoroughly assess your application with respect to factors relevant to porting:

- *Compiler dialect*
- *Hardware-dependent constructs in your code (such as word length or byte-endian dependencies)*
- *Platform run-time services*
- *Build-tool dependencies*
- *Availability of database, networking, and messaging middleware*
- *User-interface portability*
- *Test-case and test environment*

The size and complexity of the porting effort varies directly in proportion to the amount of system- and environment-dependent code. If your application uses only standard language constructs and standard libraries, it may be relatively easy to port. Java® applications, for example, usually fall into this category. If, on the other hand, your application is a C program that uses non-POSIX services on Solaris or depends on third-party products that are not available on Linux, the move can be substantially harder. Usually the system test and testing the configuration and installation of the software is an important step and takes a major part of the port.

Will my application continue to work on the original UNIX platform?

Mitigation - Even after moving development to Linux, you can continue to keep the original platform option open in order to address your other market. Porting to Linux typically means that the build tools are replaced with GNU tools, and POSIX-compliant threading libraries are used instead of the platform specific ones. GNU tools and libraries are both available on other UNIX platforms, allowing you to serve your original platform and Linux concurrently.

Porting roadmap for C and C++ applications

The following technical discussion is from a Solaris to Linux porting guide¹ (ibm.com/developerworks/linux/library/l-solar/) and might be used also for other mainstream UNIX systems.

The Linux operating system, build tools, and compilers for most languages are available free of charge and can be downloaded from the Internet. To begin the port, you will need to provide hardware and development personnel:

- *Step 1 - Download necessary development tools and Linux distribution.*
- *Step 2 - Switch over to Linux development tools while still running Solaris. This approach allows a more gradual transition.*
- *Step 3 - Become familiar with the Linux operating system while still running on Sun hardware.*
- *Step 4 - Move the application on the target Linux platform.*

Step 1: Downloads

- *Listed are the required compilers and building tools and the related online documentation links:*
- *GCC (GNU Compiler Collection) suite of tools including a C and a C++ compiler*
gcc.gnu.org/releases.html
Online documentation
gcc.gnu.org/onlinedocs/gcc-2.95.2/gcc_toc.html
- *Solaris GNU compilers and make (gmake) utility*
www.sunfreeware.com/
Online documentation
docs.sun.com/gnu.org/manual/make-3.79.1/html_mono/make.html

¹ The Solaris to Linux porting guide is an IBM document written in the Toronto Development Lab.

- *Linux can be obtained via the following Linux distributors.*

<i>Caldera</i>	www.caldera.com/
<i>Red Hat</i>	www.redhat.com/
<i>SuSE</i>	www.suse.com/
<i>Turbolinux</i>	www.turbolinux.com/

- *For Linux for UltraSPARC*
Sun www.sun.com

Step 2: Build your C/C++ application for Linux on Solaris

Set-up tools - Begin by installing the GNU tools on Solaris. Because the GNU compilers and make utility are available for Solaris as well as Linux, it is relatively easy to attempt an initial build of your Linux application using these tools.

Convert makefiles - Build your application using the GNUgmake utility instead of the Solaris make utility. Depending on the constructs used within your makefiles, you might encounter error messages generated by the GNU gmake utility as a result of differences between this and the Solaris make utility. Use the make tool documentation to help identify the problem areas within the makefiles, and adjust the makefiles as necessary.

Compile and debug - Once you have modified your makefiles to work with gmake, change the name of the C compiler being invoked from `cc` to `gcc` and the name of the C++ compiler from `CC` to `g++` within the makefiles. Then compile the application. The error messages that you encounter during the build, if any, can be classified into two categories: command-line option problems and code problems.

Identify those messages that are the result of differences in the command-line options accepted by the GNU and Sun compilers. Except for a few basic options such as “-c” and “-g”, most of the options accepted by the compilers are different. Use the compiler documentation to modify your makefiles to compensate for the differences in compiler options.

Next, you can deal with the remaining code-related error messages and warnings. Use the compiler documentation to understand and resolve the differences between the compilers.

Step 3: Build and test your application on Linux for Sun UltraSPARC

If the application being ported has a specific dependency on UltraSPARC hardware, it is an important stage in the porting process. It gives Solaris developers the opportunity to become familiar with the Linux environment without abandoning the underlying UltraSPARC hardware and without modifying the hardware-specific portions of the application.

Set-up Linux environment - Install Linux and GNU utilities on Sun UltraSPARC.

Compile and debug run-time APIs - Because you built your application using the GNU utilities, the differences you will notice in this step are limited to differences in the run-time application programming interfaces (APIs) between Solaris and Linux. See the *Run-time interfaces section* (page 7) for additional information on these differences. Make adjustments to the application as necessary.

Test - Once the application has been completely rebuilt, conduct a thorough validation test.

Step 4 : Build and test your application on target Linux platform

Set-up Linux environment - Install Linux and GNU tools on the target hardware.

Apply and rebuild application - Copy your source tree and makefiles to the new Linux machine and rebuild the application. If you did not complete Step 3, then you may need to modify the application to accommodate differences in the run-time APIs between Solaris and Linux. Finally, if your application contains any UltraSPARC-specific code, make modifications as required to resolve problems in those sections of code.

Adapt the installation routine to target Linux platform. The various Linux platforms (distributions) differ slightly in the location of configuration and startup files.

Test - Finish off the port by running a thorough suite of verification tests against the ported application. In addition test the configuration and installation on the target Linux platform.

Caution! Licenses and copyright terms - When porting to Linux and linking your application with open source libraries, be aware of the licenses and copyrights that pertain to such libraries. At a minimum, your newly ported C/C++ application will be linked to the GNU C/C++ run-time libraries. These libraries are protected by the GNU Lesser General Public License. If the software that you are porting is of a proprietary nature and you intend to sell the software once it has been built with the GNU tools, it would be prudent to fully understand the terms and conditions outlined in the GNU General Public License and the GNU Lesser General Public License (www.gnu.org/licenses/licenses.html). Certain terms and conditions in the GNU licenses will carry over to your software once it has been linked to a library protected by these licenses.

Porting roadmap for Java applications

- *Step 1 - Download necessary development tools.*
- *Step 2 - Use application on target Linux platform.*

Step 1: Downloads

Moving the development of applications written in Java from Solaris to Linux is quite straightforward. Necessary for your porting tasks are:

- Java 2 Developer Kit for Linux includes a re-engineered Java Virtual Machine (JVM) with enhanced just-in-time compiling capabilities ibm.com/developerworks/java/jdk/linux130/
- GCC suite of tools includes a Java compiler (GCJ) gnu.org/software/gcc/java/

Step 2: Use application on target Linux platform

Because the JVM accepts the same bytecodes regardless of the operating system on which the JVM runs, you have the option of compiling Java source files on one operating system and running the resulting class files on another. For example, if Linux is your primary development platform, the Java class files that you create on Linux can run equally well on Solaris or other platforms for which a compatible JVM is available.

Run-time interfaces

Although the vast majority of run-time interfaces are common between Linux and Solaris, there are some areas where differences exist. Any use your application makes of a Solaris interface not available on Linux, or not identical on Linux, will need to be modified before your application will build correctly. The areas of differences are listed below.

System calls, C library and C++ library

API differences - The Solaris kernel provides logical volume support, ACL (access control list) management for files, and system audit log capability. The Linux APIs for this functionality, where available, are not the same as those on Solaris. Also not available by default on Linux is STREAMS support. Applications that use STREAMS for networking will need to be modified to use POSIX sockets instead. Also, a number of the Solaris APIs available on Linux have different return and parameter types, or are declared in a different header file. However, these differences are often minor and typically do not require code changes in the application.

C++ library - The Solaris Forte 6 C++ Compiler includes three class libraries: complex numbers for `-compat=4`, classic iostreams, and the Standard C++ Library (which includes standard complex numbers and iostreams). GCC includes only the Standard C++ Library. If the application being ported uses the `-compat=4` complex number library or the classic iostreams library, the application will need to be modified to use the standard versions of those libraries contained in the Standard C++ Library. Making this modification has the added benefit of improving the portability of the application.

Desktop: CDE versus GNOME/KDE

Graphical user interface (GUI) - The Common Desktop Environment (CDE), which is the default Solaris desktop, is not included in the common Linux distributions. For applications that require the CDE or for users that prefer that particular GUI environment, a Linux version of the CDE is available from Xi Graphics (www.xig.com) in the form of their DeXtop product. Running DeXtop on Linux places Solaris users in a familiar GUI environment and can significantly ease the transition involved in migrating to Linux.

DeXtop requires the Accelerated-X Linux X-Server also from Xi Graphics. Although the XFree86 Linux X-Server can be made to work with DeXtop, Xi Graphics does not officially support running DeXtop on XFree86.

Threading and process management

Thread/LWP (Light Weight Process) support - Solaris supports both POSIX threads and a Solaris-specific threading model. If the application being ported uses the Solaris specific thread APIs, it would be best to modify the application to use POSIX threads instead. POSIX threads are supported on Linux and are also more portable.

Process management: /proc file system - The /proc pseudo file system provides a convenient means to access kernel data structures. On Solaris, /proc contains information about active processes and threads, and also provides an interface to control these processes and threads. The /proc control interface on Solaris is typically used by debuggers to trace program execution. On Linux, /proc does not provide a process control interface. To control processes for debugging on Linux, use the ptrace() API instead of /proc.

Operational considerations

System management: In general, managing Linux is very much the same as managing any other UNIX operating system. There are differences in the specifics of some of the commands and tasks. IBM and UNIXGuide (unixguide.net/unixguide.cgi) have published comparison charts that summarize some of the differences across not only Solaris and Linux but also AIX® and other versions of UNIX. An experienced Solaris system administrator will not encounter much of a learning curve in becoming completely comfortable with Linux.

Other third-party tools, utilities and libraries: Many popular third-party tools, utilities, and libraries commonly found on Solaris are also available on Linux, such as:

- *Rational Rose and other Rational products are available for both Solaris and Linux.*
- *The Rogue Wave SourcePro C++ library is available for Forte/Solaris as well as GCC on both Solaris and Linux.*
- *The Trolltech Qt C++ GUI framework is available for both Solaris and Linux.*

Due to the increasing popularity of Linux, the number of third-party vendors that support Linux is growing daily.

Endian format - The Sun SPARC and UltraSPARC processors store integers in big endian format. If you want to port the application to a little endian version of Linux, such as Linux/Intel, then you must resolve any dependencies on endian format within the application before the application will run correctly. As an alternative, the application can target Linux/PowerPC, Linux on zSeries, or Linux/UltraSPARC, which are all big endian platforms, instead of Linux/Intel.

Porting from Linux to Linux on zSeries

Now that the port to Linux on one platform (let's say Intel) is completed, what about porting to Linux on the other hardware platforms. The short answer is: It's "a piece of cake."

We choose zSeries as an example for two reasons.

1. zSeries is a good example of how easy a port from one Linux to any other Linux is, because the zSeries architecture is completely different from the Intel architecture. Many ISVs ported their UNIX applications to Linux on Intel first, and this is certainly a natural choice. In a second step, the porting of the application to Linux on zSeries followed without major efforts.
2. zSeries might be an interesting platform for you to support for its unique capabilities in the area of virtualization and hardware reliability and availability.

Why porting to Linux on zSeries?

Linux on zSeries is pure Linux: It is neither a Linux personality on an existing zSeries operating system, nor a special version of Linux adapted to the zSeries architecture. It has the same characteristics on zSeries that it would have on other platforms, e.g. it's a pure ASCII environment. The vast majority of the Linux structure is common to all architectures. The zSeries-dependent modifications enable Linux to communicate with zSeries memory, zSeries disk and communications hardware. The parts of Linux which interface with applications and users are unaffected.

Application advantages - Your Linux/UNIX applications on Linux on zSeries will give you the advantage of accessing enterprise data (back-end integration) which are stored on zSeries environments. This provides improved responsiveness and reduces unnecessary duplication of data. The superior capacity, scalability, reliability, availability and security of the zSeries makes it the perfect deployment platform for enterprise server applications. Linux on zSeries in addition can help to simplify operations and reduce costs by cutting the number of servers in the business environments.

Linux on zSeries porting hints and tips

These hints and tips were assembled by an IBM technical team responsible for porting applications to Linux on zSeries. The complete document is available at ibm.com/servers/esdd/articles/linux_s390/index.html

Little endian to big endian

S/390 is a big endian system. Any code that processes byte-oriented data that originated on a little endian system may need some byte-swapping. The data may have to be regenerated or, if that isn't possible (for example, shared files), the application may have to be reworked to adjust for processing little endian data. This problem does not exist, if the original platform is a big endian platform (e.g., UltraSPARC, IBM ~ pSeries™ ..).

Assembler code

Assembler code will need to be re-written in S/390 assembler. All opcodes need to be changed to S/390 opcodes. If the application code uses assembler header files, an S/390 version of the header is needed. S/390 Assembler code for Linux uses the S/390 opcodes but follows the syntax conventions of GNU assembler. You can download the GNU assembler manual at www.gnu.org/manual/gas-2.9.1/as.html

Absolute addresses and high-order bits

Some applications use hard-coded addresses for various purposes. One purpose might be to define a fixed page in memory for allocations or memory maps with `mmap()`. One form of the `mmap()` call allows for page-fixing. In this mode, `mmap()` tries to memory map using storage at the requested address. On Intel platforms, this address may be specified in a place that will not work for Linux on zSeries.

Every platform is likely to make different choices for the location of program stack, system libraries, heap, and so forth, so a hard-coded address is already a poor bet for portability to other systems. The zSeries addressing scheme also ignores high-order bits, so a hard-coded address of 0x80000000 (high-order bit on) would be translated to 0x00000000, an reserved address on zSeries. In generating addresses, arithmetic operations on addresses can also turn on high-order bits.

If an application uses an absolute address that causes a segmentation violation or other errors, it will have to be changed. `/proc/<pid>/map` shows how an active process uses storage ranges within its address space. If the code must use absolute addresses, this map can be used to find address ranges that are not already reserved.

The high-order bit for some address fields might need to be "AND'D" off (31-bit mode bit). You must do so if the address is used in arithmetic operations.

Application development tools

Deploying an application requires a robust set of tools for the target server. The Web site at ibm.com/servers/eserver/zseries/os/linux/ldt/ describes the phases involved in developing or porting C/C++ applications that will be deployed on Linux on zSeries and presents a list of tools that are useful during each phase.

Miscellaneous

ptrace and return structure - use of ptrace and the return structure is architecture dependent.

Configuration/build/Makefile scripts or files - probably need to add support for the S/390 platform.

proc file system - has some differences, e.g. :

/proc/cpuinfo	format is different
/proc/interrupts	is not implemented
/proc/stat	does not contain INTR information

Char is unsigned - Character type char is by default interpreted as unsigned char on zSeries rather than signed char as on other platforms.

Va_args - Under Linux for zSeries, a va_list definition is different from other platforms. If your application copies va_lists by simply issuing a C assignment statement, on Linux for zSeries, you must replace the copy with the _va_copy macro.

Summary

Porting from any UNIX to Linux can be easy, simple and straight-forward, particularly if your UNIX applications are written to common open standards. This paper shows advantages your application will gain on this new environment and help you to address the risks which may come along with a port to Linux. Porting your application to Linux on zSeries gives you the additional advantage of easy access to enterprise data and applications, which are stored on zSeries environments. In summation we would like to emphasize the following points:

- *Porting from any UNIX to Linux is a project which can be staged in a way that it will not impact any of your existing business. The complexity of porting varies from application to application and is determined by what programming language is used.*
- *Required skills are common skills for application developers so you should have all the skills to get started. If you need help and support, IBM provides technical support teams and porting centers around the world, that can help you to get your applications ported.*
- *The detailed porting roadmaps cover valuable hints and tips for helping to judge the complexity of porting your application. This is the base for a consistent porting plan, leading your project to a successful completion.*

The tremendous market success of Linux makes it evident that your applications should also run on Linux. Why miss the opportunity?

Porting experiences - What others are saying

Numerous articles in the press and on the Internet provide additional insight for those considering the move and porting to Linux. The effort porting applications to Linux certainly varies with the complexity of the application, programming languages used, and so forth.

ISV Sendmail (www.sendmail.com): "Sendmail, Inc. recently ported our commercial products onto IBM's Linux-based z900. The initial port took us six days to complete and performance exceeded expectations. The Sendmail, Inc. services team worked closely with IBM's knowledgeable staff, which greatly simplified the project."

ISV BMC (www.bmc.com): "When we took some UNIX-based code and ported it to Linux on the zSeries, what we found was that it wasn't really a port," says Fred Johannessen, director of strategy for BMC's Linux initiative. "We just recompiled the thing, and it worked. We told ourselves, 'It can't be this easy,' but it really was that easy."

ISV Pentaprise (www.pentaprise.de): "We ported our complete ERP system from Linux Intel to all IBM Linux hardware platforms within one day in average for each IBM platform, including support for DB2®. The organizational, technical and marketing support we got from IBM has been outstanding." -Reinhard Herrmann, CEO

SAP (www.sap.com/linux/), "Since Christmas 1999, you can run mySAP Technology on Linux for productive use in a mission-critical environment, and this is something we are really proud of. Linux is growing with incredible speed and absolutely meets the quality standards of SAP. And thanks to the platform-independent SAP architecture, the port of the SAP kernel was fairly easy. None of the SAP business applications had to be modified."

ISV TRUSTIX (www.trustix.com), a Norwegian ISV, successfully ported Trustix Xplore, a Linux Systems Management framework to Linux for S/390. "The porting to Linux for S/390 was much easier than we expected", commented the engineer after completion of the port in less than a day. With the exception of the IBM ~ iSeries™, the TRUSTIX applications are now available on all IBM servers with this porting.

ISV SICS (www.sics.se) ported a Prolog Compiler from Linux to Linux for S/390. "After the configuration-script was adapted to detect the correct machine types and installing the correct versions of required libraries and SW, the actual porting was a simple recompile and was completed in less than a day", commented the engineer who performed the port.

For more information

- *IBM's Solaris to Linux porting guide* ibm.com/developerworks/linux/library/l-solar/
- *Linux on zSeries porting hints and tips*
ibm.com/servers/esdd/articles/linux_s390/index.html
- *IBM Solution Partnership Center (SPC) porting information*
developer.ibm.com/spc/portinfo.html
- *S/390 ISV Technical Support* etpgw02.dfw.ibm.com
- *For a listing of current Independent Software Vendors (ISVs) providing applications supporting Linux for zSeries*
ibm.com/servers/eserver/zseries/solutions/s390da/linuxproduct.html
- *ISV Announcements*
ibm.com/servers/eserver/zseries/solutions/s390da/applications/appnews.html
- *PartnerWorld for Developers* developer.ibm.com/
- *Learn more about Linux on IBM xSeries, pSeries and zSeries at the IBM ~ Developer Domain.* ibm.com/servers/esdd/
- *Linux Application Tools* ibm.com/servers/eserver/zseries/os/linux/ldt/
- *Developer Products for Linux for zSeries and S/390*
ibm.com/servers/eserver/zseries/solutions/s390da/linuxproduct.html
- *Learn more about what IBM is doing with Linux in the developerWorks bi-weekly "LTC bulletin" on the Linux zone* ibm.com/developerworks/linux/, *and at the Linux at IBM Web site* ibm.com/linux
- *Browse more Linux resources on developerWorks* ibm.com/developerworks/linux/?article=lr
- *Browse more Open source resources on developerWorks* ibm.com/developerworks/oss/
- *Learn about the technical enhancements to Linux 2.4 in a paper by Linuxcare*
linuxcare.com/about-us/collateral/whitepapers/linux24-enterprise.epl.
- *Find out about 64-bit Linux at the the IA-64 Linux Project Web sites* linuxia64.org/
- *For a description of the Application Development and Deployment lifecycle, as well as lists of available tools on Linux for zSeries* ibm.com/servers/eserver/zseries/os/linux/ldt/
- *The GNU General Public License* gnu.org/copyleft/gpl.html *and the GNU Lesser General Public License* www.gnu.org/copyleft/lesser.html *and* gnu.org/licenses/licenses.html *allow anyone to view and enhance the Linux source code.*
- *GCC (GNU Compile Collection)* gcc.gnu.org/releases.html *and online documentation*
gcc.gnu.org/onlinedocs/gcc-2.95.2/gcc_toc.html
- *The GCC Suite also includes a Java Compiler (GCJ)* gnu.org/software/gcc/java/
- *GNU assembler manual at* gnu.org/manual/gas-2.9.1/as.html

- *Solaris GNU compilers and make (gmake) utility are available at sunfreeware.com/ and SUN online Product documentation's at docs.sun.com and gnu.org/manual/make-3.79.1/html_mono/make.html*
- *Access to high-end Linux systems for developers is available from the Open Source Development Lab www.osdlab.org/ and IBM's Linux Community Development System ibm.com/servers/eserver/zseries/os/linux/lcds/*
- *Third-party run-time libraries for Linux include the Rogue Wave SourcePro C++ library rougewave.com/ and the Trolltech Qt C++ GUI framework www.trolltech.com/.*
- *Java tools include Sun's Java 2 Platform Standard Edition java.sun.com/j2se/1.3/ and Forte for Java Community Edition sun.com/forte/ffj/ IDE, as well as the IBM Developer Kit for Linux ibm.com/developerworks/java/jdk/linux130/*
- *Red Hat's Solaris-to-Linux Porting Guide redhat.com/devnet/whitepapers/solaris_port/book1.html.*
- *See a comparison of system management tasks in charts published by UNIX guide unixguide.net/unixguide.cgi.*
- *Source-code management systems include GNU CSSC cssc.sourceforge.net/ and CVS cvshome.org/.*
- *Xi Graphics produces a CDE for Linux xig.com/.*
- *Gauge the commercial readiness of Linux by studying companies using Linux for significant systems. Aaxnet.com/design/linux2.html*

Learn more about Linux at IBM - ibm.com//linux/

And more specific informations about Linux on zSeries - ibm.com/zSeries/linux



Copyright IBM Corporation 2002

Integrated Marketing Communications, Server Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America

01/02

All Rights Reserved

IBM, IBM logo, the e-business logo, AIX, DB2, iSeries, PowerPC, pSeries, S/390 and zSeries are trademarks or registered trademarks of International Business Machines Corporation of the United States, other countries or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel is a trademark of Intel Corporation in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Information concerning non-IBM products was obtained from the suppliers of their products or their published announcements. Questions on the capabilities of the non-IBM products should be addressed with the suppliers.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited represent how some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.