Java Linux

java-linux@java.blackdown.org FAQ

by Stephen M Wynne

January 24 2000

This document attempts to answer the most commonly-asked questions on the *java-linux@java.blackdown.org* mailing list. Its focus is the Blackdown Java Platforms for Linux. It also serves as a form of HOW TO document for running Java on Linux. More information on running Java on Linux is available at *http://www.blackdown.org/*.

Contents

`$Id: header.sgml,v 1.14 2000/01/23 23:34:03 stevemw Exp $`

1 An Introduction to the Blackdown Project

This section offers a guide to resources provided by the Blackdown project.

> Every man, from the highest to the lowest station, ought to warm his heart and animate his endeavours with the hopes of being useful to the world, by advancing the art which it is his lot to exercise; and for that end he must necessarily consider the whole extent of its application, and the whole weight of its importance. – Samuel Johnson, April 17, 1750.

From *Pride In Work: the Samuel Johnson Sound Bite Page* quoted in number 521 at *http://www.samueljohnson.com/prideinw.html*.

1.1 Questions about Blackdown Java

This section of the FAQ discusses the Blackdown Java Linux project, mailing list etiquette, and provides pointers to other resources for Linux Java users.

Last update:

`$Id: about-blackdown.sgml,v 1.6 2000/01/15 21:52:19 stevemw Exp $`

1.1.1 What is the current status of Blackdown Java?

Please see *http://www.blackdown.org/java-linux/ports.html*.

1.1.2 How do I get started quickly?

See section 5.1.1 (How do I install the JDK?).

1.1.3 What is Blackdown Java?

The term "Blackdown Java" describes a group of Sun Java products ported to Linux by the Blackdown development team. Blackdown was the first group to port Java 1.0, 1.1, and then 1.2 to Linux. Other vendors have since followed. See section 3.5 (Other Java Linux Vendors) for an incomplete list.

If you're new to Java, please see section 7.2.1 (What is Java?) for an introduction to the Java language. If you are having any difficulty understanding terms in this FAQ, you can find definitions in Javasoft's *Glossary of Java and Related Terms* at *http://www.javasoft.com/docs/glossary.nonjava.html* or *Roedy Green's Java Glossary* at *http://mindprod.com/gloss.html*.

Here's a partial list of tools Blackdown has contributed towards porting:

- Sun's basic JDK(Java Developer's Toolkit), including:
  - JDK 1.0. See section 3.1 (Java 1.0 Edition).
  - JDK 1.1. See section 3.2 (Java 1.1 Edition).
  - JDK 1.2. See section 3.3 (Java 2.0 Edition).
  - See section 2.1 (What is the JDK?) for a description of the JDK.
- The 3D-API. See section 4.6 (Blackdown Support for the Java 3D API) for more information.
- An Internet browser Java plugin. See section 4.4 (Java Plugin).
- Java Media Framework API; See section 4.8 (JMF) for more details.

1.1.4 What information resources are available at Blackdown?

Karl Asha's Blackdown server, *http://www.blackdown.org/*, has a number of resources. Among other things, Karl provides the following pointers:

- Recent news about running Java on Linux: *http://www.blackdown.org/java-linux/info.html*
- Javasoft-supplied products: *http://www.blackdown.org/java-linux/products.html*
- Java Tools: *http://www.blackdown.org/java-linux/javatools.html*
- Information about Third-party Java products running on Linux: *http://www.blackdown.org/java-linux/otherproducts.html*

1.2 Blackdown's Philosophy

This section contains the formal statement of purpose for the Blackdown project and its participants.

Last update:

`$Id: philosophy.sgml,v 1.2 1999/12/31 08:36:20 stevemw Exp $`

1.2.1 Statement of Purpose for the Blackdown Project

> We each are dedicated to the professional development of the Java platform for Linux based on the community source concept. We see participation in the Blackdown project as a cutting-edge opportunity for intellectual cooperation between the open source community and the commercial software industry. We each are committed to abiding by the agreements we've made with Sun and other technology vendors. We aim to use their good will to further the cause of independence for software developers around the world. A bridge between the open source community and the commercial software development world is to everyone's advantage, and we would like to exemplify that relationship. We believe that the vendors with whom we partner are committed to the same ideals.

For an informal look at Blackdown's perspective on the developing "open source" relationship Sun, please see section 8.3.3 (Blackdown's Position on Sun's Community Source Policies).

1.3 The Java Linux Mailing List

This section offers assistance for those wanting to use the mailing list for the first time. The strength of the Linux Java community lies in its willingness to share information, so you are encouraged to join us no matter how new you are to Java or Linux.

Last update:

`$Id: mailing-list.sgml,v 1.6 2000/01/16 00:44:40 stevemw Exp $`

1.3.1 What topics are appropriate for the Java-Linux mailing list?

In short, anything related to Java and Linux is welcome.

Many questions will involve pleas for help, so here is a brief list of suggestions that may improve your chances for obtaining rapid assistance:

- Before you post a question asking for help, please read your distribution's Blackdown `README.*` files for important information about the build you're using.
  For example:
  1. **Java 1.2:** *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux*
  2. **Java 1.1:** *http://www.blackdown.org/java-linux/docs/faq/README/1.1/README.linux*
  3. **Java 1.0:** *http://www.blackdown.org/java-linux/docs/faq/README/1.0/Java-Linux-HOWTO.*
- Choose a descriptive **Subject:** line for your mail.
- When reporting problems, please follow the guidelines in section 6.1.8 (What Should I Report?).
- Posts with lines wrapped at 75-80 characters are easier to read!
- Spamming the list or using members' E-mail addresses for commercial purposes is forbidden.
- Some list users may ignore messages created in HTML or that include proprietary attachments.
- Do not followup improper postings to the entire list; copy *only* the sender and *Karl Asha <karl@blackdown.org>*, the list administrator.
- Please avoid inflammatory language.

We hope that the FAQ will help you solve many of your problems and enable you to participate on the mailing list more effectively. If it didn't, please contact the maintainers with your suggestions and we will improve it. See section 9.3.1 (Who maintains the FAQ?) for details.

1.3.2 How can I get subscribed? Unsubscribed?

Please don't send E-mail directly to the list requesting these status changes! Visit *http://www.blackdown.org/* and follow the links to instructions or use the links provided in the next paragraph.

The following URL would work in Netscape Mail to subscribe you: *java-linux-request@java.blackdown.org?subject=subscribe*

And this should unsubscribe: *java-linux-request@java.blackdown.org?subject=unsubscribe*

1.3.3 Where can I find archived postings from this list?

Karl Asha has arranged for the list to be archived here: *http://www.mail-archive.com/java-linux@java.blackdown.org/*.

1.4 Other Resources for Linux Java Users

Information about resources other than Blackdown for Linux Java users.

Last update:

`$Id: other-resources.sgml,v 1.3 2000/01/11 06:50:22 stevemw Exp $`

1.4.1 Where can I find more information and Java tools for Linux?

- Levente Farkas maintains Java RPMs and Java-Linux related links: *http://anna.inf.u-szeged.hu/java/java-rpm.html*
- Freshmeat sometimes mentions new Java developments: *http://freshmeat.net/*
- There's a GNU Java page: *http://www.gnu.org/software/java/java.html*.
- A strong newcomer is *How to do Java Development with Linux* at *http://www.jlinux.org/*.

2 About Blackdown Java for Linux

This section provides a brief overview of using the Blackdown Java platform on Linux, including an introduction to the JDK itself, a synopsis of system requirements for various editions of Blackdown Java, and how to download the bits.

2.1 Questions about the JDK

This section introduces the core Java Developer's Toolkit to Java newcomers.

Last update:

`$Id: what-is-jdk.sgml,v 1.3 2000/01/16 00:44:43 stevemw Exp $`

2.1.1 What is Sun's JDK?

Sun's JDK , or Java Developer's Toolkit, is a complete (if Spartan) development environment supporting the Java programming language. If you're new to Java, please see section 7.2.1 (What is Java?) for an introduction to the Java language.

The JDK includes the basic tools needed for developing and running Java applications, including (among other things) the following:

- A Java Virtual Machine, or JVM. The virtual machine interprets and executes Java byte codes (a more condensed, intermediate form for Java instructions) inside a software-only machine. In other words, the JVM emulates a hardware platform, including registers, program counter, and so forth.
- A Java compiler, which takes a text file containing Java syntax and transforms it into Java byte codes.
- An appletviewer for executing Java applet code.

- A Java debugger. Java 2 adds the JPDA, or "Java platform debugging architecture" to improve debugging.
- Example Java programs.
- The Java class library, including platform-independent library source.
- A tool for creating and managing Java security keys.
- A remote method stub and skeleton generator.
- A registry server for remote method invocations.
- C and C++ headers for extending the JVM with interfaces to native code.
- Native libraries for the platform-dependent portions of the JVM, including AWT layer above Motif and X11.
- Native libraries for embedding a JVM in other native applications.
- In Java 2, a JIT (just in time compiler) is available. JITs can speed performance because they compile Java byte codes into native executable regions of memory.

Note: consecutive versions of the JDK have historically included more APIs than the previous ones. See *http://www.javasoft.com/products/* for more detail. Documentation for Java APIs is available separately from Javasoft at *http://www.javasoft.com/docs/*.

For certain Blackdown distributions, there are static and dynamic versions of the core Java binaries, as well as versions compiled with and without debugging symbols. See section 4.7 (Support for the Java Platform Debugging Architecture) for more information about debugging Java.

For information on the Blackdown JDK porting project itself, including who the major contributors have been, please see section 9.2.1 (Who Ported the Blackdown JDK?) .

2.1.2 Can I commercially redistribute the JDK?

According to our contacts at Sun, no. You are allowed to redistribute the JRE. See section 2.2 (Questions about the JRE) for more information. You may also want to see section 8.3 (Java Licensing and the Open Source Community) for the latest information about this topic. Of course, you should always contact Sun yourself if you have any questions related to your business plans.

2.2 Questions about the JRE

This section introduces the JRE, or Java Runtime Environment to Java users or developers who either need a smaller Java runtime for their applications or only want to ship the smallest JVM possible when they deploy their code as a complete application.

Last update:

`$Id: what-is-jre.sgml,v 1.3 2000/01/16 00:44:44 stevemw Exp $`

2.2.1 What is Sun's JRE?

With Java 1.1, Sun needed a way to distribute a JVM without shipping tools for building Java classes. The JRE includes only what is required for running Java applications, including a Java Virtual Machine, the core API class libraries, and a set of default properties.

2.2.2 Can I commercially redistribute the JRE?

According to our contacts at Sun, yes. You may wish to contact Sun directly for case by case issues.

2.3 An Overview of Using Blackdown Java

This section provides an overview of using Java on Linux, describes the currently-supported version of the Blackdown JDK, discusses what hardware is supported, and addresses the question of whether or not Sun will support Java commercially. It's not easy to keep this section updated, so please bear with us.

Last update:

```
$Id: overview.sgml,v 1.10 2000/01/24 03:45:41 stevemw Exp $
```

2.3.1 What can I expect from Blackdown Java on Linux?

You should be able to run most "pure Java" (see Sun's *http://java.sun.com/100percent/* page) applications on the Linux JDK or JRE. It has tended to take longer to get the latest JDK ported to a target (Linux) that Sun didn't officially support. Javasoft and Inprise are jointly working to provide commercial support for Java on Linux now. See section 2.3.5 (Is Javasoft Going to Support Java on Linux?) for more information about Sun's announcement to support Java on Linux.

For getting up and running with the Blackdown Java on your system as soon as possible, see section 5.1.1 (How do I install the JDK?).

2.3.2 What is the currently-supported version of Blackdown Java?

At the time of this writing, the latest supported Blackdown JDK version is **1.1.8**. Both Blackdown and Inprise have "RC" (release candidate) distributions of 1.2.2 at the time of this writing.

Please see section 2.5.1 (Where do I obtain JDK 1.02, 1.1, and 1.2?) for places to download.

To determine your version, use this command:

```
$ java -version
```

Keep in mind that other Java virtual machines (JVMs) exist, so check for a `README.linux` file in the root directory of the distribution to ensure that it's Blackdown's.

2.3.3 For what other hardware architectures is Blackdown Java available?

Please see section 5.8 (Questions about Non-x86 Blackdown JDK Ports) for more information. Also, visit the Blackdown site and check on the latest status of a given platform at *http://www.blackdown.org/java-linux/ports.html*.

With Java 2, it's been very difficult for Blackdown developers to keep up with the APIs and rapid development pace of Javasoft. Intel is currently the first target to get completed in any typical Blackdown development effort. More than ever before, Java 2 has showed the need for a more standardized relationship between Sun and Blackdown. We continue to ask for clarification on the developer recruiting process, code and bug exchange schedule, and for formal rules for new developer Java Compatibility Kit (JCK) access. For more on these issues, please see section 8.3.3 (Blackdown's Position on Sun's Community Source Policies).

2.3.4 Under what license is Blackdown Java available?

Standard disclaimer: none of this is legal advice.

- You can download and use the Blackdown JDK or other products from any of the Blackdown mirrors.
- You are only allowed to redistribute the JRE.

Please see section 8.3 (How do I understand Java licensing?) for more information.

2.3.5 Is Javasoft going to support Java on Linux?

Finally, yes. Sun is working with Inprise to outsource portions of this project. The December 7th, 1999 Inprise press-release is available at *http://www.inprise.com/about/press/1999/sunlinux.html*. It leads off with this statement:

> Inprise Corporation (Nasdaq: INPR) today announced that it has jointly produced a Linux version of the Java(tm) 2 Platform, Standard Edition (J2SE), with Sun Microsystems. The agreement enables programmers to develop and run applications based on the Java 2 platform and deploy them on Linux workstations and servers, as well as simplify moving existing Java applications to the Linux operating system (OS).

2.3.6 Where can I find online reactions to the Inprise release?

In this December 9th, 1999 article *http://dailynews.yahoo.com/h/zd/19991209/tc/19991209113.html* by ZD-net reporter Steven J. Vaughan-Nichols, he states:

> It was a press conference like any other press conference. Sun announced that, in conjunction with Inprise, it was releasing Java 2 Platform, Standard Edition (J2SE). What Sun didn't mention, though, was that much of the work that lay behind J2SE was done by the Linux Blackdown Porting Team, a group of open-source programmers devoted to Java.

An article at *http://www.computerworld.com/home/news.nsf/all/9912083sunlinux* by Computer World reporter by David Orenstein quotes Blackdown's Karl Asha who said,

> "As the Linux community goes, I suspect they'll follow the most stable, most recent, and best-performing JDK [Java Development Kit], regardless of who produces it."

and in a December 8, 1999 press release at *http://www.javasoft.com/features/1999/12/linux.html* Sun acknowledges,

> Blackdown has made significant advancements to the availability of Java technology for Linux," said Gina Centoni, director of product marketing for Sun Microsystems. "Sun would like to thank Blackdown for its ongoing efforts in delivering Java technology on Linux. We look forward to continued collaboration with Blackdown with the common goal of providing the Java 2 platform to the Linux community."

And in *JINI-USERS posting #69 on December 8th 1999* in the thread *Re: Sun releases "its" JDK for Linux - a warning for SCSL licensees*, Ken Arnold said:

> This matters to me, of course, because this stuff with the JDK will affect people's ideas of what we will do with the Jini technology. I can't promise no screwups, but I'm with Dick Gabriel on this one – If I see Sun stealing credit for other people's work at this level, both my resignation and protest will be in the same letter. We will give credit where it's due. I think people who have seen us in action (for example, at the Jini Community Summits) have seen this happen.
>
> I apologize for Sun's mistakes here, and promise that we will do our best to avoid them in the future, especially on the Jini project where we have the most control.

For more background on the Linux community's long-held hopes for full Sun support for Java, see section 9.1 (History of the Blackdown Java Effort).

2.4 Blackdown Java for Linux System Requirements

Discussion of hardware and software requirements for Blackdown Java.

Last update:

```
$Id: requirements.sgml,v 1.2 2000/01/16 00:44:42 stevemw Exp $
```

2.4.1 What are the minimum system software requirements?

Although Blackdown developers compile and distribute Java x86 binaries compiled on RedHat 6.1, and that is considered to be the reference Linux distribution, the Linux kernel, C library, and X11R6 distribution are the essential requirements.

- A Linux operating system kernel:
  - JDK 1.1.x: Kernel 2.0.37.
  - JDK 1.2: Kernel 2.2.12-20.
- To run AWT, you'll need X11 up and running; most users run an XFree86 X11 display server. See *http://www.xfree86.org/* for details.

  - XFree86-3.3.5-3.
  - XFree86-xfs-3.3.5-3 (for Java 2 TrueType font support).

- The appropriate C library:
  - JDK 1.1.8: GNU glibc-2.0.7-7.
  - JDK 1.2.2: GNU glibc-2.1.2-11.
  - JDK 1.02 runs with libc-5.3.12-31 on recent systems.

RedHat package name/versions are given above. More information about the GNU C library is found in section 7.1.2 (What is glibc, and Why Does it Matter?). See section 6.4.1 (Why are TrueType fonts required for Java 2?) for more about TrueType.

So RedHat 6.1 is the reference Linux distribution? Yes. This is also briefly discussed in section 5.4.2 (Is RedHat 6.1 the reference distribution for Blackdown Java?).

2.4.2 What are the minimum hardware requirements?

It depends on which version you run, but here are some actual numbers taken from a RedHat 6.1 x86 system. Alpha, SPARC, and PPC platforms may have different requirements.

- A supported platform running regular, 32-bit Linux is required. Blackdown recommends a Pentium-class CPU at a minimum.
- You'll need a display card if you want to run X11 and AWT. You may want to look into devices that support 3D, Mesa, and Linux at *http://www.mesa3d.org/*. See section 4.6.2 (3D installation under RedHat 5.2) for more information about setting up 3D with Blackdown Java.
- JDK 1.02
  1. **Memory**
     `VSZ (k) .................................. 23,504`
     `RSS (k) ................................... 4,376`
  2. **Disk**
     `(k) ...................................... 10,432`
- JDK 1.1.8v1
  1. **Memory**
     `VSZ (k) .................................. 26,720`
     `RSS (k) ................................... 8,468`
  2. **Disk**
     `(k) ...................................... 40,091`
- JDK 1.2.2 RC3
  1. **Memory**
     `VSZ (k) .................................. 82,264`
     `RSS (k) .................................. 17,600`

2. **Disk**
   `(k) ....................................... 61,243`

- Sound cards are supported but not required.

The system used to measure these examples was a 64MB P133 running RedHat Linux 6.1 with a uniprocessor build of kernel 2.2.12-20; the above numbers were produced by the `ps(1)` command while running the ImageMap demo with appletviewer.

If you have the memory requirements suggested above, and you still have fatal errors related to running out of memory, see section 6.1.5 (Help, Java Said it Ran out of Memory!) for suggestions of flags to pass to the JVM for setting memory and stack allocation limits. Java uses a *lot* of memory, so be prepared to specify additional RAM in your application deployment recommendations.

What about my particular hardware platform? See section 5.7 (Questions about x86 Blackdown Java Ports) for information about x86 platforms (including Intel and clone builds). See section 5.8 (Questions about Non-x86 Blackdown Java Ports) for other targets.

2.5 Downloading Blackdown Java for Linux

Help with downloading Blackdown Java.

Last update:

`$Id: download.sgml,v 1.9 2000/01/15 21:52:19 stevemw Exp $`

2.5.1 Where do I obtain JDK 1.02, 1.1, and 1.2?

Please visit *http://www.blackdown.org/java-linux/mirrors.html* and select the site nearest you. Both Randy Chapman's JDK 1.02 and Steve Byrne's JDK 1.1.x are found on participating mirror sites. Check section 2.3.2 (What is the currently-supported version of Blackdown Java?) for the currently supported version the Blackdown JDK.

2.5.2 How do I install?

See section 5.1.1 (How do I install the JDK?).

2.5.3 What is the currently-supported version of Blackdown Java?

2.5.4 Which is the latest version?

For the latest status on a given JVM, check the *Java-Linux Ports* page at *http://www.blackdown.org/java-linux/ports.html*

2.5.5 OK, which version of the JDK should I download?

This section describes the organization of the Blackdown distribution mirror hierarchy. You may also be interested in reading section 2.4.1 (What System Configuration do I Need to Run the JDK?) before you decide what to download.

After reading this, if you're still wondering which individual archive to download, please see 2.5.7 (What are These Files?) below.

2.5.6 JDK 1.02 Downloads:

This version of the Blackdown JDK is only supported on Intel. If you know of other Linux platforms to which the JDK has been ported, please tell the FAQ maintainer. Look for the mirror closest to you by following instructions in section 2.5.1 (Where to obtain JDK 1.02, 1.1, and 1.2?).

2.5.7 JDK 1.1 Downloads:

JDK 1.1.7 was the latest release to compile on all platforms. You may need to visit *http://www.linuxppc.org/java/* in order to obtain the PPC version. Intel and ARM have support in 1.1.8. For the latest on Java 1.1, see section 3.2 (Information relevant to the Blackdown 1.1 JDK).

Here's a view of the 1.1.7 directory:

- **Alpha**
  .... `alpha/`
- **Intel x86**
  .... `i386/`
- **SPARC**
  .... `sparclinux/`
- **PowerPC**
  .... `mklinux/`
- **Patches to Javasoft Source, etc...**
  .... `common/`

You may wish to read section 7.1.2 (What is glibc, and Why Does it Matter?) to help you understand which Intel JDK to download.

For an official description of the FTP site, look for an up to date version of *http://www.blackdown.org/java-linux/docs/faq/README/FTP*, at the top level of your closest mirror.

To just run the JDK, you don't need anything out of `common/`. Those **diff(1)** files are for people who want to request free access to the **SPARC/x86 Solaris** and **i386 Win32** sources from Sun and help with the port. See section 8.4.1 (Compiling the Blackdown JDK from Source) for more information.

What are these files in the FTP mirror directories? By now, many readers are still wondering what each archive in the mirror directories is. Below is a representation of what your favorite web browser would have displayed if you were visiting a Blackdown JDK mirror recently.

```
Current directory is
/pub/linux/devel/lang/java/blackdown.org/JDK-1.1.7/i386/glibc/v3
    README.linux
    i18n_1.1.7-v3-glibc-x86.tar.bz2
    i18n_1.1.7-v3-glibc-x86.tar.gz
    jdk_1.1.7-v3-glibc-x86-native.tar.bz2
    jdk_1.1.7-v3-glibc-x86-native.tar.gz
    jdk_1.1.7-v3-glibc-x86.tar.bz2
    jdk_1.1.7-v3-glibc-x86.tar.gz
    jre_1.1.7-v3-glibc-x86-native.tar.bz2
    jre_1.1.7-v3-glibc-x86-native.tar.gz
    jre_1.1.7-v3-glibc-x86.tar.bz2
    jre_1.1.7-v3-glibc-x86.tar.gz
    rt_1.1.7-v3-glibc-x86-native.tar.bz2
    rt_1.1.7-v3-glibc-x86-native.tar.gz
    rt_1.1.7-v3-glibc-x86.tar.bz2
    rt_1.1.7-v3-glibc-x86.tar.gz
```

As a hypothetical user, you've already decided that your system is based on `glibc` (perhaps by reading 7.1.2 (What is glibc, and Why Does it Matter?)). Now you'd like to know what these four files do for you, and which ones you should download.

1. **jdk**. The JDK contains everything you need to run Java, compile Java, and debug java. It doesn't contain the international character converters (see i18n).

2. **jre**. The JRE is the Java runtime environment, including the international character converters.

3. **rt**. The Java RT is the minimal configuration you can have to run Java. It doesn't contain any internationalization support.

4. **i18n**. This contains font mappings and a JAR file containing the international character converters, and really doesn't need to be split off by library type, but is anyway.

5. ***native***. These are additional binaries that provide support for native threads (according to command-line options and environment variables).

For more about Blackdown Java 1.1, see section 3.2 (Information relevant to the Blackdown 1.1 JDK.), although most 1.1 information is distributed throughout the FAQ.

When do I need extra *native* binaries? All versions of Blackdown Java up through 1.1.7v3 required you to take the extra step of installing a package of shared libraries containing native the threads VM if you wanted to effectively run with the `-native` flag. Blackdown Java 1.1.8 does not require you to unpack anything special. See the section 4.2.3 (Is there a native threads port of Blackdown Java?) for a little more information.

2.5.8 JDK 1.2 Downloads:

At this time, only the `sparclinux` and `i386` directories were populated. Internationalization is included, as well as native threads support. For more about Blackdown Java 2, see section 3.3 (Information relevant to the Blackdown 2.x Java Ports).

How should I download Java archives ending in .sh? These are files that consist of a shell-script at the top and a ZIP file containing the binary archive of the Java product you wish to download. Please use binary mode.

How do I deal with downloaded files ending in .bz2? These are bzip2 compression arcvhies. Please see *The bzip2 and libbzip2 Official Home Page* at *http://sourceware.cygnus.com/bzip2/* for availability of precompiled **bunzip2(1)** executables (as well as source code). However, you should have this on any contemporary distribution of Linux.

Hey, why doesn't Blackdown use xdelta? Several of you have requested that we supply *XDELTA* binary patches in order to save download time. At the time of this writing, no one had time to consider doing this extra step. However, we're looking for volunteers to help with development, and if you were willing to work on this sort of thing, it would be helpful.

For more information about XDELTA (it *is* cool), feel free to visit *http://www.xcf.berkeley.edu/~jmacd/xdelta.html*.

3 The Various Blackdown Java Editions for Linux

Discussion of the various Blackdown builds of JDKs, Java APIs, and related Blackdown-maintained tools for using Java on Linux.

3.1 Information relevant to the Blackdown 1.02 JDK.

Discussion of issues related to the first Blackdown JVM to be widely available on Linux, version 1.02.

Last update:

```
$Id: jdk-1.0.sgml,v 1.1.1.1 1999/12/27 06:13:31 stevemw Exp $
```

3.1.1 What do I need to know about the Blackdown JDK 1.02?

It still works! Actually, if you have details about using this build in your daily work, please report them to the FAQ maintainer.

See section 9.2.1 (Who Ported the Blackdown JDK?), which credits Randy Chapman, the one who got it all started (with encouragement from Karl Asha).

3.2 Information relevant to the Blackdown 1.1 JDK.

Limited discussion of issues related to Blackdown 1.1 JVMs. You'll find that most 1.1 material is present throughout the FAQ since this incarnation of it (after Eric S. Raymond's) originated during an intense period of Blackdown JDK 1.1 development.

Last update:

```
$Id: jdk-1.1.sgml,v 1.4 2000/01/06 11:35:55 stevemw Exp $
```

3.2.1 What is the latest 1.1 release? Does it support native threads?

The last release to support all platforms was 1.1.7v3. It supports native threads via command-line flags. The latest release for Intel and ARM was 1.1.8 at the time of this writing.

See the section 4.2.3 (Is there a native threads port of Blackdown Java?) for a little more information.

3.2.2 What about all those versions of 1.1?

In short, if you are on Intel, use 1.1.8 (or later if available by the time you read this). On other architectures, the situation is different. Here is a small compatibility chart:

- **glibc-2.0 systems** require JDK 1.1.7v1a and earlier.
- **glibc-2.1 systems** require JDK 1.1.7_v3 and later.

See 2.5.5 (OK, which version of the JDK should i download?) for which versions are supported on what platforms.

3.2.3 Where can I get SWING (JFC) for Java 1.1?

See *http://java.sun.com/products/jfc/*

3.2.4 Is there a JIT for Blackdown Java 1.1?

Not from Blackdown, but see section 3.4 (Just In Time Compilers for Blackdown Java) for other sources.

3.3 Information relevant to the Blackdown 2.x Java Ports

Discussion of running the latest version of Java from Blackdown.

Latest update:

```
$Id: jdk-2.0.sgml,v 1.7 2000/01/15 21:52:21 stevemw Exp $
```

This section describes Blackdown's 2.0 port of Java, currently being termed "Java 2, Standard Edition," or J2SE by Sun.

3.3.1 When is the JDK 1.2 Going to be Released for Linux?

The Blackdown team has had the J2SE sources for over a year, and has been actively working on porting it. Starting March 5th, 1999 several J2SE "release candidates" (RCs) have been issued and are available through the Blackdown mirrors. Visit *http://www.blackdown.org/java-linux/mirrors.html* and download the latest.

Furthermore, *http://www.inprise.com/* now has its own offering. See 2.3.5 (Is Javasoft going to support Java on Linux?) for more information.

3.3.2 What about the libc compatibility and Blackdown Java 2?

At this point, you're best-off with a `glibc-2.1.2-11` system (in RedHat 6.1 parlance) because new builds and tests are currently being done only against that library, either on Debian or RedHat. It's not clear that builds *couldn't* happen for older libraries; they just aren't.

3.3.3 Will there be a patch for the latest Java 2 release candidate?

Juergen Kreileder <kreilede@issan.cs.uni-dortmund.de> determines this on a case by case basis. For example, 1.2.2 RC4 will have been issued with a set of "diffs," despite a requirement for users to have the 1.3 sources and combine the 1.2 and 1.3 trees. Stay tuned to the mailing list. See section 8.4.10 (What is a source patch?) for the definition of a patch.

3.3.4 How do I upgrade from libc2.0 to libc2.1 ?

Currently, glibc2.1.2 is required to run recent builds of Java 2 from Blackdown. Look for information about upgrading at your favorite distribution's website, but you may also discover answers the FSF (Free Software Foundation's) GNU libc web site: *http://www.gnu.org/software/libc/libc.html*. If you are on RedHat 6.1, you already have it installed. For information specific to your platform, including RedHat, see section 5.2 (Linux Distribution-specific Questions).

3.3.5 What does "Standard Edition" in the J2SE acronym mean?

With Java 2, Sun is splitting the Java product out into three targeted environments:

1. **Standard Edition** (J2SE)
2. **Enterprise Edition** (J2EE)
3. **Micro Edition** (J2ME)

The standard edition is the core JVM with class libraries. The enterprise edition has some of the other (apparently pure Java) APIs glued into it including enterprise Java beans. The micro edition is a range of options for supporting a smaller Java.

Blackdown Java 2 development is currently limited to the "standard edition." See Sun's *Java 2 Platform* page for more about the three different environments and J2SE documentation: *http://java.sun.com/java2/*.

3.3.6 Does Swing (also know as JFC) come with the JDK 1.2?

Yes, the Java Foundation Classes come with the J2SE.

3.3.7 Is Sun's sunwjit just in time compiler working with JDK 1.2.2?

As of 1.2.2 RC3, sunwjit JIT compiler does still not work entirely correctly, according to *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux.*

3.3.8 Is Sun's HotSpot JIT being ported?

According to *Juergen Kreileder <kreilede@issan.cs.uni-dortmund.de>* on December 29th, 1999, he had begun work on this project.

See Javasoft's *Java HotSpot Performance Engine* page at *http://www.javasoft.com/products/hotspot/* for more information about the HotSpot JIT (just in time compiler).

3.3.9 What about the new debugging interfaces for Java 2?

Please see section 4.7 (Support for the Java Platform Debugging Architecture).

3.4 Just In Time Compilers for Blackdown Java

This section discusses various "just in time" compilers (JITs) and Blackdown Java on Linux.

Last update:

```
$Id: jits.sgml,v 1.4 1999/12/31 08:36:19 stevemw Exp $
```

3.4.1 Is there a JIT (Just in Time Compiler) for the Blackdown JDK?

There are several. Please take a look at Shudo Kazuyuki's references in his *Just In Time compiler for Sun Java Virtual Machine / x86: shuJIT* page at: *http://www.shudo.net/jit/#refs* for a comprehensive list of pointers to JIT-related projects.

There are several others including these open-source projects:

- **ShuJIT**, Shudo Kazuyuki's contribution: *http://www.shudo.net/jit/.*
- **TYA**, by Albrecht Kleine <java@ak.sax.de>: *ftp://gonzalez.cyberus.ca/pub/Linux/java/.*
- **LaTTe:** *http://latte.snu.ac.kr/.*

See section 3.3 (Information relevant to the Blackdown 2.x Java Ports) for more information about new features in Blackdown Java 2.0 builds, including its default JIT.

3.5 Other Vendors' Java Editions for Linux

This section describes a few of the vendors now supporting the Java platform on Linux. It can never be as up to date as we would like, but it can serve as a starting point if you're looking for more information.

Last update:

`$Id: other-vendors.sgml,v 1.2 2000/01/04 16:08:11 stevemw Exp $`

3.5.1 What other vendors and individuals are supplying Java for Linux?

This section of the FAQ is incomplete. Please visit *http://www.blackdown.org/java-linux/productlist.html* for more detail.

- Inprise has a JVM and a JIT that run on Linux. See *http://www.inprise.com/* for more detail.
- IBM has several JVMs that run on Linux. See *http://www.ibm.com/developer/java/* for more detail.
- Japhar, an LGPL-based project. See *http://www.japhar.org/* for more detail.
- Kaffe, a GPL-based project. *http://www.kaffe.org/* for more detail.

4 Blackdown-supported Java APIs

Discussion related to the plethora of Java APIs, only some of which are currently supported on Linux.

4.1 The Gamut of Javasoft APIs Versus the Blackdown Ports

This section answers questions about APIs (application programming interfaces) supported by various Blackdown ports.

Last update:

`$Id: blackdown-apis.sgml,v 1.4 2000/01/06 11:35:55 stevemw Exp $`

4.1.1 What Java APIs are supported?

Each Blackdown Java edition tries to support whatever were the core APIs for that version of Java. For those details, see section 3 (The Various Blackdown Java Editions for Linux). One of the biggest problems in Java 1.1 were threads, the interface to which consists of an API in and of itself.

Here are a few of the APIs which are getting a lot of attention. We'll add more information here when we have time.

- See section 4.2 (Threads and Blackdown Java) for information about the threading API and problems related to Linux and threads.
- For the Jini connection API, see section 4.3 (Blackdown Support for Jini).

- To find out more about the JVM extension for Netscape, see section 4.4 (Blackdown Java Browser Plugin).
- The JNI API is discussed in section 4.5 (Blackdown Support for Java Native Interfaces).
- Support for Java's 3D API is treated in section 4.6 (Support for the Java 3D API).
- Talk about Java's new platform debugging architecture is found in section 4.7 (Support for the Java Platform Debugging Architecture).
- 4.8 (Support for Java Media Framework API).

4.1.2 What is the latest status on a given API?

For the latest status on a given API, check the *Java-Linux Ports* page at *http://www.blackdown.org/java-linux/ports.html*

4.2 Threads and Blackdown Java

This section discusses various issues with the java.lang.Thread *API, an area with a reputation of portability problems for Java. Blackdown Java on Linux has had its share of gotchyas.*

Last update:

```
$Id: threads.sgml,v 1.8 2000/01/11 06:50:23 stevemw Exp $
```

4.2.1 What are the two types of threads used in Java on Linux?

- Native threads. These are one-to-one threads, or threads with a user space API backed by the kernel itself.
- "Green" threads. These are user level threads without explicit kernel support. For more on these, see 4.2.4 (What are Green threads?).

4.2.2 Are native threads on Linux limited in any way?

Yes, because the Linux kernel provides only a one to one mapping, the thread model doesn't scale as far as hybrid user and kernel space threading models such as LWPs on Solaris. However, one could argue that the Solaris model itself is excessive, and that reasonable applications should not expect to support thousands of threads at once.

For a rough idea of how many threads your version of the kernel can support per JVM, consider the total number of tasks your kernel has been compiled to handle, and what your hardware can actually support. Here's a canned identifier search in the latest stable kernel for `NR_TASKS` using the *Linux Cross-Referencer*: *http://lxr.linux.no/ident?i=NR_ TASKS*.

4.2.3 Is there a native threads port of Blackdown Java?

Yes, in both 1.1.7 and 1.2.x. Phill Edwards added native threads support to 1.1.7 support; the 1.1.8 native threads implementation has been back-ported from 1.2.2. See section 2.5.7 (When do I need extra *native* binaries?) if you're wondering which distribution requires you to unpack `j*-native.tar.*` files.

You can find more information about related command-line options and environment variables in the *http://www.blackdown.org/java-linux/docs/faq/README/1.1/THREADS-117v1a* and *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux* files that accompany binary Blackdown Java distributions.

4.2.4 What are Green threads?

Green threads use some magic with `jmp_buf`s, stack pointers, the program counter, and some other hocus pocus to schedule different instruction streams. This gives the practical effect of multi-threading with some caveats about signals and how they're handled across `longjmp()`s. See the discussion of native threads in section 4.2.3 (Is there a native threads port of Blackdown Java?).

4.2.5 What about Green threads and console priority?

According to *Michael.Sinz@sinz.org*,

> As of JDK 1.1.6 v5, Kevin fixed the STDIO (System.in/System.out/System.err) problem. In JDK 1.1.7 V1 it was fixed even better for the poll() changes.

Previous discussion of this problem is included for people on older JVMs.

Jon Wätte <hplus@zilker.net>, has a brief synopsis of Green threads at *http://www.zilker.net/~hplus/jvm.html*. He states:

> Sun's Java VM as running on Solaris [without the native threads package] used a user-level thread package named Green Threads. This is strictly a collaborative threading package; the JVM is one system process and only executes one Java code stream at a time. This is convenient for implementors, because special operations that required nobody else to touch sensitive data are intrinsically protected. As long as you don't relinquish the CPU (and block signals) you're in no danger.
>
> If you do a [System.in.readLine()], the JVM will sit in the blocking input call until input is available; nothing else will happen in your program.

Bernhard Bablok <bablokb@lrz.uni-muenchen.de> has this suggestion:

> [The] `System.in.readLine()` [method] won't block other threads, if their priorities differ. My experience is, it doesn't matter which threads have a higher/lower priority. Priority shouldn't be much lower than `Thread.NORM_PRIORITY` (much lower means two lower on Linux).

4.2.6 Is there a problem with Green threads and the JVM invocation API?

Yes, there's a known problem with Javasoft's user-level threads package (called Green threads) and the Invocation API because of the way it wraps system calls. Note that there is native thread support in the current JDK. See section 4.2.3 (Is there a native threads port of Blackdown Java?).

Green Threads and Wrapped System Services Uncle George, Juergen Kreileder, and Bernd Kreimeier have encountered this problem, and through their discussion of it we have this list of references to consult for more information:

- JDC Bug Parade bug ID at *http://developer.java.sun.com/developer/bugParade/bugs/4040743.html*
- *The Java Tutorial* under "Invoking the Java Virtual Machine" at *http://java.sun.com/docs/books/tutorial/native1.1/implementing/*
- Rob Gordon's excellent *Essential JNI* published in 1998 by *Prentice Hall PTR*, ISBN 0-13-679895-0

Gordon suggests in *Essential JNI* on page 260 that:

> When building a Java-enabled application on Solaris and linking against the Green threads version of libjava.so, you must defer system calls until after the call to JNI_CreateJavaVM. This is because the user-level threads package in libjava.so (Green threads) defines its own system calls . . .

And the *JDK 1.1 Tutorial* suggests that, "The first step toward getting this to work is always to link libjava.so into the executable *first*, even before libc.so . . ."

Hacking Around the System Service Wrappers On June 21, 1998 *Uncle George <gatgul@voicenet.com>* posted these paraphrased comments:

> Actually, library calls are overridden, not system-calls. If you are savvy enough, you can call the system service yourself, and forget the library interface (i.e. instead of using select, try `__select`). But in doing so, the green threading . . . can be compromised, i.e. other threads won't be runnable until that service completes. The **malloc()/free()** bogus pointer messages are problems in trying to free a pointer twice, of which some "MALLOC" packages get terminally upset with . . .

You may also be interested in the discussion of native threads in section 4.2.3 (Is there a native threads port of Blackdown Java?)

4.2.7 Where can I find more information about threads on Linux?

See the *Linux Threads Library* page at *http://pauillac.inria.fr/~xleroy/linuxthreads/* for FAQs, documentation, and source for the native threads API used in Blackdown Java.

4.3 Blackdown Support for Jini

This section discusses Blackdown support for Jini, a distributed API intended for "ubiquitous computing" with Java-enabled devices.

Last update:

`$Id: jini.sgml,v 1.2 1999/12/31 08:36:19 stevemw Exp $`

4.3.1 Does Blackdown Java support the Jini API?

Yes, but you need to be using Java 2, preferably a recent build.

On September nine, 1999, Blackdown developer *Cees de Groot <cg@cdegroot.com>* had this to say:

I'm using it and it works. What's more to say?

If you are having problems, check three things:

1. Multicast support in the kernel (>=2.0.37);
2. The correct multicast route in your routing table (dest=224.0.0.0 mask=240.0.0.0 dev=eth0);
3. No multicast routing support in the kernel (I have the suspicion that this interferes with looping back local mc packets that are leaving the box).

I have seen RMID crashing after running a while on the JDK 1.2 pre-release builds, you probably want to disable the JIT for the RMID (this could also be due to the fact that I'm running the RMID on my SMP box, though).

4.3.2 Where can I find out more about Jini?

- See Javasoft's page on the web *http://www.javasoft.com/products/jini/*.
- You can browser the JINI-USERS mailing list at *http://archives.java.sun.com/archives/jini-users.html*.
- The Jini community website (put up by Sun as an outreach to community source users) is at *http://www.jini.org/*.

4.4 Blackdown Java Browser Plugin

This section describes Blackdown support for the Internet browser "plugin" for adding a given JVM to Netscape.

Last update:

`$Id: browser-plugin.sgml,v 1.3 2000/01/06 11:35:55 stevemw Exp $`

4.4.1 Is there a browser plugin for standard Java support?

Blackdown now has access to the source for the 1.2 version of the plugin, has made an initial RC (release candidate), and intends to continue to develop it. See the *Java-Linux Ports* page at *http://www.blackdown.org/java-linux/ports.html* for its current status.

4.4.2 Where can I read more about Sun's Java plugin products?

See *http://www.javasoft.com/products/plugin/*.

There is a FAQ page at Javasoft, as well: *http://www.javasoft.com/products/plugin/plugin.faq.html*.

4.4.3 What needs to be done to add plugin support to HTML pages?

Amit Patel <amitp@theory.stanford.edu>, who has been working on HTTP proxies answers:

The Java Plug-in requires that web page authors make changes to their existing HTML code if they wish to have their JDK 1.1-based applets run using the Java Plug-in, rather than the browser's default Java runtime. While the changes are not trivial, they are not overly complex either.

To make this process easy for web page authors, Sun provides the Java Plug-in HTML Converter, free of charge, to automate this process. In addition, Sun has provided a detailed specification outlining the HTML changes required to use the Java Plug-in, and how web page authors can implement them "by hand". The Java Plug-in HTML Converter can be downloaded from the HTML Converter download page. For the HTML specification, see the Java Plug-in HTML Specification.

4.4.4 What does the Java Plug-in HTML Converter do?

Again, *Amit Patel <amitp@theory.stanford.edu>*, answers:

The Java Plug-in HTML Converter provides both a default template (default.tpl), and three supplementary templates. These supplementary templates allow web page authors to more explicitly target the browsers and platforms used in their environment when modifying their pages using the Java Plug-in HTML Converter:

1. default.tpl: The default template used by the Java Plug-in HTML Converter. The converted page can be used in Internet Explorer and Navigator to invoke the Java Plug-in on the supported platforms.

2. extend.tpl: The converted page can be used in any browser on any platform. If the browser is Internet Explorer or Navigator on Windows 95, Windows NT, or [some UNIX-like OS such as Linux], Java Plug-in will be invoked. Otherwise, the browser's default Java runtime is used.

3. ieonly.tpl: The converted page can be used to invoke Java Plug-in in Internet Explorer on the supported platforms.

4. nsonly.tpl: The converted page can be used to invoke Java Plug-in in Navigator on the supported platforms.

4.5 Blackdown Support for Java Native Interfaces

This section describes Blackdown support for accessing native code through the JNI API.

Last update:

```
$Id: native-interfaces.sgml,v 1.9 2000/01/11 06:50:21 stevemw Exp $
```

4.5.1 How does JNI work with Blackdown Java?

This section is under construction. Recent mailing list traffic would indicate that there are major C library transition issues affecting JNI portability and general functionality.

- **JDK 1.0.** Presumably the older JNI API worked, but no historical information was forthcoming at the time of this writing.
- **JDK 1.1.** See section 3.2 (Information relevant to the Blackdown 1.1 JDK.) for the latest updates.
- **JDK 1.2.** Again, see section 3.3 (Information relevant to the Blackdown 2.x Java Ports).

4.5.2 Why should I be interested in Native Interfaces on Linux?

- Code written well for Linux can be portable across most BSD and SYSV-derived systems.
- There are times when having the source for your JVM is invaluable when you are developing a JNI.
- Debugging your problem can be easier with an entirely open-source toolchain.
- Affordable hardware and operating system software.
- There's a large community of dedicated developers.

4.5.3 Is there anything different about native interfaces on Linux?

Not really. The compilation flags may need to be different from what Javasoft recommends in the Solaris-oriented documentation.

Juergen Kreileder <kreilede@issan.cs.uni-dortmund.de> points out that with the new Native Threads package, you should add the `-D_REENTRANT` flag to your other options. Pointers to more information about the native threads pack is available in section 4.2.3 (Is there a native threads port of Blackdown Java?).

Keep in mind that there has been a known problem with all green threads implementations and the JVM Invocation API, discussed in section 4.2.6 (Is There a Problem with Green Threads and the JVM Invocation API?).

There have been reports of problems with C++ *IOStreams* using `libstdc++-2.7.x`. Note that `g++-2.8.1` and `libstdc++-2.8.1` have been tested, and seem to work fine for several users on RedHat 5.0.

Compiling Library Code for JNI We've provided a couple of rudimentary examples (including Makefiles) below. They all assume that you have the JDK installed in `/usr/local/java/jdk`. These worked with both JDK 1.1.8 and 1.2.2RC3 on RedHat 6.1 i386, with the `egcs-1.1.2-24` and `egcs-c++-1.1.2-24` packages installed.

- For a small C example, see: *http://www.blackdown.org/java-linux/docs/faq/examples/jni-in-C/*
- And for a small C++ example, see: *http://www.blackdown.org/java-linux/docs/faq/examples/jni-in-C++/*
- And for a small C++ Invocation example (JDK 1.2.x only), see: *http://www.blackdown.org/java-linux/docs/faq/examples/invocation-in-C++/*

Examples like these can be found in chapter two, *Getting Started* of Sheng Liang's *The Java Native Interface: Programmer's Guide and Specification* in the *Java Series* published by Addison Wesley in 1999 (ISBN 0201325772). You can find more elaborate examples at the book's web page here: *http://java.sun.com/docs/books/jni/*.

4.5.4 How can I step into native methods used by the JVM?

When you run `java_g`, the environment variable `$DEBUG_PROG` can be used to specify a debugger to use. For more about this topic, pick up a copy of Rob Gordon's *Essential JNI* published in 1998 by *Prentice Hall PTR*, ISBN 0-13-679895-0 and look at chapter 14, "Debugging Native Methods."

4.5.5 Where are some good online references for JNI on Linux?

- David Dagon has a partially-complete reference for JDK 1.1.7 on Linux at: *http://www.mindspring.com/~david.dagon/jni/Native.txt*. Note: this was last dated November 30, 1998.

4.6 Blackdown Support for the Java 3D API

This section describes Blackdown support for the Java 3D API.

Last update:

```
$Id: 3d.sgml,v 1.1.1.1 1999/12/27 06:13:31 stevemw Exp $
```

4.6.1 Does the Java 3D API work with Blackdown Java?

Although the information in the FAQ may be outdated by the time you read it, it was still under development at the time of this writing.

Java 3D requires Java 2. Second, Blackdown has only built it for i386 (at the moment). Work continues. See the *JDK 1.2 Status Page* at *http://www.blackdown.org/java-linux/jdk1.2-status/* for the latest, as usual.

To read more about Java 3D, see Sun's website at: *http://www.javasoft.com/products/java-media/3D/index.html*.

4.6.2 How do I install the 3D package if I'm on RedHat 5.2?

Kenneth W Flynn <flynnk@rpi.edu> posted this explanation:

3D installation under RedHat 5.2

1. Download and install the Java3D package as suggested: `java3d1_1_1pre-v1-linux-sdk_tar.bz2`.
2. If you do not already have them, you will also need the Mesa 3D Graphics Library at *http://www.mesa3d.org/*.
3. Download Version 3.0, and compile it. Copy the libraries from `.../Mesa-3.0/lib` to `/usr/lib` or your favorite library location that `ldconfig` can find.

4. Do an `ldconfig -v` to make those libraries available.
   The following is also needed to get Java working, but as a newbie, I had to go hunting for this information, so it bears repeating. These workarounds are listed on the known-bugs page: *http://www.blackdown.org/java-linux/jdk1.2-status/known-bugs.html*.

5. You will need the URW fonts as suggested on the Blackdown homepage. Do a Save As on the link marked 'here' on the following page: *http://www.gimp.org/fonts.html*.
   Unbundle this software, and copy it to your X11 URW fonts directory, which for me is:

   ```
   cp URW/* /usr/lib/X11/fonts/URW/
   ```

   You also need to update your XF86Config file's font section to match the one shown on the webpage; then restart X.
   I had many of the fonts here, but I just did the bulk copy to make sure I had all of them. I guess you could pick and choose if you were bored.

6. You will also need to create a symbolic link to the libstdc++ library.

   ```
   ln -s /usr/lib/libstdc++.so.2.8.0 /usr/lib/libstdc++-libc6.0-1.so.2
   ```

Hopefully, Java 3D now works. :)

4.7 Support for the Java Platform Debugging Architecture

Discussion of Blackdown support for Java Platform Debugging Architecture (JPDA) API, as well as general debugging issues for Java on Linux.

Last update:

```
$Id: jpda.sgml,v 1.3 2000/01/06 11:35:56 stevemw Exp $
```

4.7.1 Is the JPDA supported with recent Blackdown JVM 1.2 builds?

Basic JPDA support has been added with 1.2.2. With RC4, full support will be available, including JPDW (the Java debug wire protocol) and JDI (the Java virtual machine debug interface). We hope to add more documentation specific to Linux about this soon, but for now, please visit Sun's site at *http://www.javasoft.com/products/jpda/*.

4.7.2 How can I use a debugger to examine native methods used by the JVM?

See section 4.5.4 (How can I step into native methods used by the JVM?), which may offer some help.

4.8 Blackdown Support for Java Media Framework API

This section describes Blackdown support for the JMF, or Java Media Framework API.

Last update:

```
$Id: jmf.sgml,v 1.1.1.1 1999/12/27 06:13:32 stevemw Exp $
```

4.8.1 Is the Java Media API ported to Java on Linux?

Blackdown developers have been working with Sun to stay current with JMF sources, We hope to add more information about this soon, but for now, please see

http://www.blackdown.org/java-linux/jdk1.2-status/ for more information.

To learn more about the JMF in general, Sun's site is at: *http://www.javasoft.com/products/java-media/jmf/*.

4.8.2 Is there a pure-Java implementation of the JMF?

On September eight, 1999, *Amith Yamasani <amith@eng.sun.com>* had this to say:

You can use the JMF1.1 All-Java version on Linux. There is no separate Linux build and the Solaris build cannot be used on Linux. The all-java version does not support some formats that require native libraries (such as Indeo, MPEG and so on). Please see the JMF 1.1 documentation page for a list of supported content-types and codecs.

You can also try out the JMF 2.0 beta release (all-java) version. It has support for more codecs than JMF 1.1 and includes a pure java MP3 decoder.

[Amith claims you can obtain these files at *http://java.sun.com/products/java-media/jmf/*.]

5 Running Blackdown Java on Linux

This section discusses aspects of running Blackdown Java on various Linux distributions in detail. Because Blackdown developers have tended to use RedHat and Debian exclusively, mailing list participants are encouraged to contribute as much as they can to this section.

5.1 Blackdown Java Installation Questions

Discussion related to installing Blackdown Java on various versions of Linux.

Last update:

```
$Id: install.sgml,v 1.2 1999/12/30 12:01:39 stevemw Exp $
```

5.1.1 How do I install the JDK?

This section addresses installs for JDK 1.1.x, but the same principles apply on other versions.

1. Download the **tar(1)** archive for your platform as directed by the mirror list mentioned in section 2.5.1 (Where do I obtain JDK 1.02, 1.1, and 1.2?).
2. Move to an appropriate directory and for example, unpack the **tar(1)** archive as follows:

```
$ cd /usr/local
$ tar zxvf /tmp/jdk1.1.6.tar.gz
```

3. Make sure your DISPLAY environment variable is set, put the resulting directory into your execution path, and try something:

```
$ export DISPLAY=:0
$ export PATH=/usr/local/jdk1.1.6/bin:$PATH
$ appletviewer \
    /usr/local/jdk1.1.6/demo/awt-1.1/lightweight/OpenlookButtons/example.html
```

Note that the above examples assumed you were using the Bash shell, such as `/bin/bash`. Also note that if you already have a `CLASSPATH` set, you'll probably want to append `.:/usr/local/jdk1.1.6/lib/classes.zip:` to that environment variable. This ensures that you have the current working-directory in your class search path, and makes things much easier for you as you test your code.

Of course you may want to add these environment commands to your shell's startup script, such as `$HOME/.bashrc`.

Some people may wish to set `JAVA_HOME` themselves. That way, you can change all commands following setting that to use it instead, such as

```
$ export JAVA_HOME=/usr/local/jdk1.1.6
$ export PATH=$JAVA_HOME/bin:$PATH
$ export CLASSPATH=.:$JAVA_HOME/lib/classes.zip
```

To verify your install, use this command:

```
$ java -version
```

5.1.2 Why doesn't Blackdown supply .deb or .rpm packages?

It's simply a lack of time. Volunteers are welcome to roll and offer their own.

5.2 Linux Distribution-specific Questions

Discussion related to specific Linux distributions.

Last update:

`$Id: dist-specific.sgml,v 1.4 2000/01/05 18:13:34 srh Exp $`

5.2.1 What distributions of Linux support Blackdown Java?

Although we currently have more information about running Blackdown Java on RedHat Linux for x86, many other platforms are running the bits. Please contribute what you know to the FAQ.

- Debian. See section 5.3.1 (Blackdown Java on Debian). (Please complete!)

- RedHat. See section 5.4 (Blackdown Java on RedHat).
- Slackware. See section 5.5 (Blackdown Java on Slackware). (Please complete!)
- SuSE. See section 5.6 (Blackdown Java on SuSE). (Please complete!)

5.2.2 I'm on <your favorite Linux distribution here> what do I need?

Missing libXp There are some problems that have been common to various platforms of Linux. From the README file:

> The JDK now depends on having the X printing library, libXp installed with the rest of your X files. This is part of the most recent X11 distributions, so you'll need to upgrade (or, if you are daring, you can just extract the libXp.so* file from the upgrade package for your Linux distribution and install it by hand w/o upgrading everything else, although this is unlikely to work if your X libraries are substantially out of date).

The glibc versus libc5 Problem The latest versions of the Blackdown JDK 1.1.x have their own built-in loader (libdl.so) and libc.so, which will hopefully make them more robust in most cases. Older versions of the JDK 1.1.5 (especially) are much more sensitive. Take a look in `$JAVA_HOME/i386/green_threads/` to see these files. (You may remove them if you feel they are a source of trouble to you.) Karl Asha maintains a list of JDK-related library issues at: *http://www.blackdown.org/java-linux/docs/libraries.html*. In short, you need `glibc-2.0.7` and `glibc-devel-2.0.7` or higher on a glibc-based system such as RedHat 5.0 or Debian 2.0. There are RPMs for these on RedHat Intel systems at *ftp://ftp.redhat.com/pub/redhat/current/updates/i386* or *ftp://ftp.redhat.com/pub/redhat/updates*.

For more information about Blackdown Java on RedHat, see section 5.4 (Blackdown Java on RedHat Linux).

For a `libc5` system, you need `libc.so.5.44` or greater, along with a recent `libdl.so`. You can find these at *ftp://tsx-11.mit.edu/pub/linux/packages/GCC*.

For an in-depth discussion of this issue, see section 7.1.2 (What is glibc, and Why Does it Matter?).

5.3 Blackdown Java on Debian GNU/Linux

This section will hopefully include information about running Blackdown Java on Debian GNU/Linux.

Last update:

```
$Id: debian.sgml,v 1.1.1.1 1999/12/27 06:13:31 stevemw Exp $
```

5.3.1 Does Blackdown Java work on Debian?

Yes, but you need to be running a version that has the libraries you need.

For now, please see *http://www.debian.org/* for more information.

You're welcome to contribute what you know to the FAQ.

5.4 Blackdown Java on RedHat Linux

This section provides information about running Blackdown Java on RedHat Linux, the most popular distribution of Linux to date.

Last update:

`$Id: redhat.sgml,v 1.2 2000/01/16 00:44:42 stevemw Exp $`

5.4.1 Does RedHat Linux support Blackdown Java?

It depends on which release and which version of a particular package, but yes, several of the developers run Blackdown Java on their RedHat systems.

5.4.2 Is RedHat 6.1 the reference distribution for Blackdown Java?

Yes, RedHat 6.1 is currently the reference distribution for all Blackdown Java products, at least on x86. Other architectures will inevitably involve exceptions.

5.4.3 RedHat Blackdown Java Installation Tips

If you're on RedHat 5.2 and older, download the `glibc` and `ld` updates described under *http://www.redhat.com/support/docs/errata.html*. In any case, it is advisable to stay as current as possible with the RedHat errata.

RedHat RPM Availability According to *Sorin Lingureanu <sorin@tuiasi.ro>*, JDK RPMs are to be found in *ftp://ftp.redhat.com/pub/contrib*. Levente Farkas has more information about running the Blackdown JDK on RedHat here: *http://anna.inf.u-szeged.hu/java/java-rpm.html*.

RedHat/RPM Environment Variable Issues In addition to the general problems described in section 6.1.1 (Environment Variables), make sure that you don't have conflicting `JDK_HOME`, `JAVA_HOME`, and `CLASSPATH` environment variables set from scripts sourced from `/etc/profile.d/` at login. This can especially be a problem when trying multiple versions of the JDK in succession.

5.4.4 What do I need to do in order to run Blackdown Java 1.2.x?

We assume you've read section 2.4 (Blackdown Java for Linux System Requirements) and you know you need to upgrade your system.

You'll have to be running RedHat 6.x. The RedHat 6.1 i386 upgrade mechanism worked for the FAQ editor, who was running an updated version of RedHat 5.2.

1. Back up your system.
2. Boot off the RedHat CD or floppies.
3. Follow instructions to update your OS and related packages. Select the option to install the 5.2 compatibility libraries. This may (or may not) help with older Java packages.

4. Check your font path. RedHat 6.x uses the font server, but your old `/etc/X11/XF86Config` file will continue to use the filesystem. Make sure your font path includes the contents suggested in the new `/etc/X11/fs/config` file. (See section 5.4.6 (What's wrong with my font server?) for more information.

5. Update your system with the latest goodies from *ftp://updates.redhat.com* (see the RedHat errata mentioned in section 5.4 (RedHat Blackdown Java Installation Tips)).

5.4.5 Does RedHat support TrueType Fonts?

Yes, as of 6.0. Please see *http://www.redhat.com/support/manuals/RHL-6.0-Manual/install-guide/manual/doc012.html*. For more information about TrueType on X11, see 6.4.3 (Will the Situation with TrueType Fonts Get Better?).

5.4.6 What's wrong with my font server?

If you've chosen to "upgrade" your RedHat system to 6.x, the font server, and hence, TrueType fonts, won't work by default. You want to follow the instructions for working around this found in the *Redhat Install Guide* mentioned above. Here's the explanation of what to do from the section "Miscellaneous New Features."

> Please Note: Those of you who upgrade will not have this feature until you edit your fontpaths. To do this you must edit the /etc/X11/XF86Config file. Scroll down until you see fontpaths listed. Replace them all with FontPath "tcp/localhost:7100". You must also verify that xfs, the X Font Server, is running. By issuing the command /sbin/chkconfig –add xfs you will insure that it starts at system boot time.

5.5 Blackdown Java on Slackware Linux

This section will hopefully include information about running Blackdown Java on Slackware Linux, one of the earliest distributions.

Last update:

```
$Id: slackware.sgml,v 1.1.1.1 1999/12/27 06:13:32 stevemw Exp $
```

5.5.1 Does Blackdown Java Work on Slackware?

For now, please see *http://www.slackware.com/* for more information.

You're welcome to contribute what you know to the FAQ.

5.6 Blackdown Java on SuSE Linux

This section will hopefully include information about running Blackdown Java on SuSE Linux, a distribution with strong commercial support from Europe.

Last update:

```
$Id: suse.sgml,v 1.2 1999/12/30 12:01:41 stevemw Exp $
```

5.6.1 Does Blackdown Java work on SuSE?

For now, please see *http://www.suse.com/* for more information.

You're welcome to contribute what you know to the FAQ.

5.7 Questions about x86 Blackdown Java Ports

Information about ports to processors compatible with the x86 instruction set. For now, most x86 information is strewn around the FAQ, but as issues clearly in this category arise, we will put them here.

Last update:

```
$Id: x86.sgml,v 1.2 1999/12/31 10:50:59 stevemw Exp $
```

5.7.1 Intel-specific Issues

Intel (i386) processor port information.

Last update:

```
$Id: i386.sgml,v 1.3 1999/12/31 10:50:57 stevemw Exp $
```

Nothing for this section yet.

5.7.2 AMD-specific Issues

AMD processor port information.

Last modified:

```
$Id: amd.sgml,v 1.2 1999/12/31 10:50:56 stevemw Exp $
```

Nothing at this time.

5.7.3 Cyrix-specific Issues

Cyrix processor port information.

Last modified:

```
$Id: cyrix.sgml,v 1.2 1999/12/31 10:50:57 stevemw Exp $
```

Nothing at this time.

5.8 Questions about Non-x86 Blackdown Java Ports

This section attempts to provide some additional information about non-Intel versions of the Blackdown JDK. At this time, Karl Asha has a more complete catalog of the different platforms available at http://www.blackdown.org/java-linux/ports.html. *Help determining which JDK to download is available in section 2.5.5 (OK, which version of the JDK should i download?).*

Last update:

`$Id: non-i386.sgml,v 1.5 1999/12/30 12:01:40 stevemw Exp $`

5.8.1 Alpha Issues

Alpha processor port information.

Last modified:

`$Id: alpha.sgml,v 1.1 1999/12/29 07:53:30 stevemw Exp $`

Is there an Alpha Port? Yes, there have been (several 1.1 builds). Please see *http://www.blackdown.org/java-linux/ports.html* for more information. Uncle George, *Uncle George* <*gatgul@voicenet.com*>, also has Alpha ports for download at *http://www.voicenet.com/~gatgul/Java.html*.

Unfortunately, there isn't any current Alpha information at the time of this writing.

5.8.2 Arm Issues

Arm processor port information.

Last modified:

`$Id: arm.sgml,v 1.2 1999/12/30 12:01:39 stevemw Exp $`

What is the status of the Arm ports? On December 28th, 1999, *Scott Hutinger* <*srh@www.mprojects.wiu.edu*> offered this informal status report:

> Arm is in limbo currently.

He did say that with time and resources, Blackdown developers would like to pursue the following:

- Running 1.2 JCK on the 1.1.8 port looking for glibc-related problems.
- Working on getting some kind of a JIT up and running for 1.2.

5.8.3 PPC Issues

PowerPC processor port information.

Last update:

`$Id: ppc.sgml,v 1.3 1999/12/31 10:50:58 stevemw Exp $`

Is there a PowerPC Port? There has been for some time. Please see *http://khendricks.ivey.uwo.ca/linuxppc/jdk/* for more information.

5.8.4 SPARC Issues

SPARC processor port information.

Last update:

```
$Id: sparc.sgml,v 1.6 2000/01/06 11:35:56 stevemw Exp $
```

What's the status of the SPARC Port? *Note: this section needs updating.*

Our Java 1.1 information is rather dated, but see section 2.3.2 (What is the currently-supported version of Blackdown Java?) for details about the latest supported build.

On June 16, 1998 *Johan Vos <johan@sesuadra.org>* provided this summary of the current SPARC effort, which is based at *http://tinukai.sesuadra.org/~jdk/*:

> Starting from JDK 1.1.5, the Intel and SPARC ports are based on the same code. The SPARC-specific issues are separated from the other linux-specific issues. Improvements and bug-fixes in the general source code can be implemented without having to re-implement the SPARC-specific code. As a consequence, most problems and updates on the Intel-JDK also occur on the SPARC-JDK. I try to synchronize as much as possible with Steve [Byrne]'s ports – although licensing problems caused the JDK 1.1.6v1 release for SPARC to be a few days after the Intel-release.

Johan follows up on 11/09/1998 with this comment:

> [The] JDK for sparclinux systems currently only exist for libc 5 systems. I am working on a port for glibc, but this will be a jdk 1.2 port. . . .

What is the status of the Blackdown Java 2 for SPARC? On December 28th, 1999, *Eddie C Dost <ecd@skynet.be>* wrote:

Sparc does want to support HotSpot, as there is no other JIT available for Sparc Linux at current. So far things look quite good. Porting to i86 should be ok, other archs are harder to get.

Sparc is doing quite well, there where some kernel problems with floating point on Ultra Sparc, and some kernel related signal problems, but these are fixed in the latest 2.2.x and 2.3.x kernels. With these Sparc happily passes the JCK so far at least on RedHat systems (other platforms should be tested soon, problems expected are minor).

Does a SPARC JIT exist for 1.2? On December 28th, 1999, *Scott Hutinger <srh@www.mprojects.wiu.edu>* pointed out that code for a SPARC JIT 1.1 does exist, although it's Solaris specific, pointing us at the *LaTTe: A Fast and Efficient JVM JIT* page at *http://latte.snu.ac.kr.*

He mentioned that Blackdown SPARC developers are interested in LaTTe as well as the HotSpot JIT from Sun. See section 3.3.8 (Is Sun's HotSpot JIT being ported?) for the latest information on general HotSpot support in Blackdown Java.

6 Troubleshooting Blackdown Java on Linux

This section attempts to answer your initial questions when you're trying to solve problems with Blackdown Java.

6.1 Troubleshooting and Reporting Bugs in Java on Linux

This section offers help for people trying to help themselves deal with Blackdown Java when it's broken.

Last update:

`$Id: solving.sgml,v 1.8 2000/01/11 06:50:22 stevemw Exp $`

6.1.1 What should I consider when troubleshooting Java on Linux?

- **Does your system hardware and software satisfy the minimum requirements?** See section 2.4.1 (What System Configuration do I Need to Run the JDK?).
- **Are you running the current version?** You can find that information in section 2.3.2 (What is the currently-supported version of Blackdown Java?).
- **Have you checked to see if your libc, glibc, and X11 libraries could be the problem?** See section 5.2 (Linux Distribution-specific Questions). If you learn something we can add to there, please post them!
- **Environment Variables.** Here are the top candidates for problems:
  - `CLASSPATH`
    Should either be unset or start with `.:$JAVA_HOME/lib/classes.zip`.
  - `PATH`
    Should include `$JAVA_HOME/bin` – ahead of `/usr/bin` where the *Kaffe JVM* often lives, *please!*
  - `JAVA_HOME`
    This should be set to your top-level JDK directory.
  - `JDK_HOME`
    Although the JDK doesn't need this, it ought to be set to your top-level JDK directory (if at all). HotJava uses this environment variable, as do other packages other than the JDK.
- **Are all the required X11 fonts installed?**
  This is a subtle area of concern. People have experienced crashes from not having 75dpi fonts installed.
- **Are your font and localization properties correct?**
  Another area that is difficult to troubleshoot. I hope to add more information here soon!

For more information about the Kaffe JVM, see *http://www.kaffe.org/*.

6.1.2 Kaffe and the Blackdown JDK 1.1 are Installed. Neither Work!

If you have the *Kaffe JVM* installed, you may see something like this:

```
javac HelloWorld.java
```

and seeing this message:

```
Failed to open object 'HelloWorld/java'
```

Other people have reported error messages such as these:

```
$ javac
Failed to locate native library in path:
        /usr/lib
Aborting.
```

Daniele Lugli <danlugli@tin.it> reports:

```
 ...The javac which was giving the problem was /usr/bin/javac,
that is the kaffe javac. Now that I moved $JAVA_HOME/bin
ahead in my $PATH, both javac and java work, and I see a
wonderful "Hello, World!".
```

Always check for conflicting environment variable problems. For details on how to do this, see section 6.1.1 (Environment Variables).

A very common cause for problems with Kaffe is having `/usr/bin` in your path ahead of `$JAVA_HOME/bin`.

6.1.3 I Have the Blackdown JDK 1.1.x and RMI is Giving Me Fits!

Rich McClellan <richmc@entropy.ucsc.edu> reports that running the Blackdown **JDK 1.1.5v7libc** works better than the glibc version, and forwards this advice from *Alexander V. Konstantinou <akonstan@cs.columbia.edu>*:

```
The problem is related to naming in /etc/hosts. This is a
known Java problem in picking up the right name for the
interface. Change the localhost name to your real machine
name. This may cause problems with talk though ...
```

6.1.4 I've upgraded to ld.so-1.9.9 on Slackware and now Java coredumps!

Slackware JVM 1.1.x users have found that removing the `libdl.so.*` and `libc.so.*` files from `$JAVA_HOME/lib/i386/green_threads` takes care of this problem. We don't know what causes it yet, though.

6.1.5 Help, Java said it ran out of memory!

You have two strong candidates as causes for this problem.

1. Your program needs more than the default that the JVM allows. Memory limits can be set like this: `java -ms16m -mx64m ....` From the JDK documentation:

   ```
   -ms<number>       set the initial Java heap size
   -mx<number>       set the maximum Java heap size
   ```

2. You actually don't have enough virtual memory (the total of physical RAM and swap space) available. The `free` command can help you there. See the manual page for **free(1)**. For the current minimum system requirements, see the discussion of them in section 2.4.1 (What System Configuration do I Need to Run the JDK?).

6.1.6 How do I add swap space?

See the manual pages for **swapon(8)** and **mkswap(8)**.

6.1.7 Where can I report bugs?

Karl Asha has setup a *Jitterbug* problem-report form at *http://www.blackdown.org/cgi-bin/jdk*. Please report problems by filling this in as best you can. There are some guidelines for this in section 6.1.8 (What Should I Report?).

Before submitting a bug, you may be able to help by visiting Javasoft's Bug Parade at *http://developer.java.sun.com/developer/bugParade/bugs/* and checking for related problems. If you mention a bug from a JDC bug ID in your problem report, it can help Blackdown JDK developers. At this time, you will have to join the JDC in order to search this database.

As usual, feel free to open a discussion about your problems on the *java-linux@java.blackdown.org* mailing list.

6.1.8 What should I report?

Although it may not seem so, solving Linux JDK problems can sometimes be difficult because of the large variation of systems that are in use. Please do your best when reporting a bug to tell us all about your system, the code you're running, and anything else that can help. Here's a checklist of items to mention.

1. A careful synopsis of the problem.
2. Which JDK you're using. Type `java -version` to see.
3. Your Linux distribution (i.e. RedHat) and its version.
4. The LibC version you're using. Type `ldconfig -D 2>&1 | grep libc | tail -1` to see.
5. The loader you're running. Type `ldconfig -D 2>&1 | grep ld | tail -1` to see.
6. *XFree86 Version*. Type `xdpyinfo | grep 'release number'` to see.
7. The version of the Linux kernel you're running. Type `uname -r` to see.
8. *JDC BugID* if you've found one.
9. A URL where with some related Java code or binary data.
10. Your assessment of the problem's priority.

6.1.9 Are there any debugging techniques I can try myself?

Try running the JDK java, appletviewer, and javac wrapper scripts like this:

```
$ sh -x $JAVA_HOME/bin/java -verbose -verbosegc -cs -verify ...
```

or this:

```
$ strace -f -s 256 $JAVA_HOME/bin/java ...
```

That last command will be *very* noisy, but you may be surprised what you learn from reading it.

6.1.10 That didn't help. What else can I do?

We've created a troubleshooting information collection script that you can download from *http://www.blackdown.org/java-linux/docs/faq/tools/jsysinfo.sh* and follow the instructions at the top of it. Read the output carefully; it's full of hints that can help you solve the problem on your own.

For serious troubleshooters, see the advice given in section 4.5.4 (How can I step into native methods used by the JVM?).

6.2 Blackdown Java and System Environment Variables

This section covers questions about Blackdown Java's interaction with environment variables on the system.

Last update:

```
$Id: envs.sgml,v 1.4 1999/12/31 16:16:05 kreilede Exp $
```

6.2.1 What environment variables affect Blackdown Java?

On June 7th, 1999 *Nelson Minar* <*nelson@media.mit.edu*> put this list together.

The following is documentation for environment variables the JDK supports. Note, this is not a comprehensive document for all variables: generic variables like LD_LIBRARY_PATH and LOCALE are not mentioned unless they have Java-specific meanings.

1.1 refers to 1.1.7_v3a, 1.2 refers to 1.2pre-v1. The Blackdown ports support both Green threads and native threads - some variables are only valid in one of the two.

Variables with Blackdown specific behaviour Java 1.1 and Java 1.2:

```
Variable            Settings
--------            --------
THREADS_FLAG        "green" for green threads
                    "native" for native threads
```

Variables with Blackdown specific behaviour Java 1.1 and Java 1.2, green threads only:

```
Variable            Settings
--------            --------

JDK_NO_KERNEL_FIX   "true" to remove fix for kernel accept() bug

DO_NOT_FREE         Turn off freeing memory entirely.

DO_NOT_CHECK_MEM    If set, removes some checks for double-freed memory
                    bugs associated with AWT. Dangerous, but slightly
                    faster.
```

Variables with Blackdown specific behavior Java <= 1.1.7v3 only, all threading models:

```
Variable            Settings
--------            --------
NS_JAVA             If set, runs a JDK with no AWT or X11. Starts up
                    faster, no graphics. See also DISPLAY

DISPLAY             If unset, acts like NS_JAVA is set (1.1.7v3 only).
                    See also man page X(1)

DYN_JAVA            If set, Motif is dynamically linked instead of
                    using the Motif statically linked into the JDK bins.
```

Generic JDK variables, Java 1.1 and Java 1.2:

```
Variable            Settings
--------            --------
CLASSPATH           See JDK documentation

JAVA_COMPILER       If set to "tya", then "libtya.so" will be used as
                    a JIT compiler if it is installed correctly. If
                    set to "none", no JIT will be used.
                    Java 1.1 uses no JIT if unset
                    Java 1.2 on i386 uses the Sun JIT (sunwjit) if unset
```

6.3 Blackdown Java and X11

This section discusses display problems and their workarounds with Blackdown Java.

Last update:

```
$Id: x11.sgml,v 1.1.1.1 1999/12/27 06:13:33 stevemw Exp $
```

6.3.1 Can I run Java without X11 installed?

Michael Sinz of the Blackdown team explains:

> In versions of the JDK 1.1.x that predate JDK 1.1.7v3 you need to set the NS_JAVA environment variable.
> As of JDK 1.1.7v3, the JDK will also support noticing that you do not have the DISPLAY environment set and thus will use the NS_JAVA setting.

6.4 Font Problems and Blackdown Java

This section discusses font problems and their workarounds on Blackdown Java.

Last update:

`$Id: fonts.sgml,v 1.5 2000/01/16 00:44:37 stevemw Exp $`

6.4.1 Why are TrueType fonts required for Java 2?

They're needed for Java2D. Also, UNICODE (see *http://www.unicode.org*) mapping is only correct with TrueType fonts.

6.4.2 Can't Blackdown just copy fonts from someplace into a Distribution?

Blackdown developer *Juergen Kreileder <kreilede@issan.cs.uni-dortmund.de>* answers:

I've asked some time ago and was told that it is not possible because of copyright. The copyright is somewhat strange, *http://www.softseek.com/Graphics_and_Drawing/Fonts/Fonts/Review_17322_index.html* says:

> The font is technically not freeware – Microsoft has secured from The Monotype Corporation the rights to distribute these fonts from their Web site at no cost.

The MS site is *http://www.microsoft.com/typography/fontpack/default.htm*. So we would have to ask Monotype. The fonts and the EULA are available from *http://www.agfamonotype.com/soft_dev/* (click on Euro).

6.4.3 Will the situation with TrueType fonts get better?

Juergen Kreileder <kreilede@issan.cs.uni-dortmund.de> answers again:

The problem is that XFree86 doesn't support TrueType fonts yet, this will get better with XFree 4.0.

Juergen continued by saying that upcoming releases of Java 2 from Blackdown would have improved tweaks to `fonts.properties` to help them to work more smoothly.

On January 5th, 2000, Juergen explains further about the missing zapf dingbats and symbol problem:

> The problem is that the URW fonts are available with two different foundries: urw and abode. The two foundries use different family names for the fonts too. The reason why we currently use the urw foundry is that the abode foundry uses ambiguous family names, this may result in using the symbol fonts from {75,100}dpi instead of the symbol font from urw, this gives a bad font display. So the best way to use the urw fonts with the JDK currently is to use a fonts.dir file with urw foundry and a font.alias file with the adobe foundry. Doing it vice versa is not

possible as the JDK currently can't deal with font.alias files. [I've attached the fonts.* files from the Debian gsfonts-x11 package.]

Also, it's advisable to make the fonts available through the Xserver directly (i.e. do not use a font server for the urw fonts). This allows the JDK to get direct access to the font files.

6.4.4 Why don't we just turn off the warnings?

The general consensus among experienced Java Linux developers is that to not warn users of an actual deficiency would be a mistake. Simple workarounds exist, however.

6.4.5 Will there be future support for TrueType Fonts in X11?

Yes, and FAQ editor's note: you can also find more information about displaying TrueType on X11 from *http://www.darmstadt.gmd.de/~pommnitz/TrueType/xfsft.html* and *ftp://ftp.xfree86.org/pub/XFree86/current/doc/RELNOTES* or go to the *XFree86 Home Page* at *http://www.xfree86.org/* and follow the links to *Current Release*.

6.4.6 Where can I learn more about TrueType in X11?

See *http://www.linuxdoc.org/HOWTO/XWindow-User-HOWTO.html*.

6.5 Problems with Window Managers and Blackdown Java

This section discusses problems with running Blackdown Java with X11 window managers. Also see section 6.3 (Blackdown Java and X11).

Last update:

`$Id: window-managers.sgml,v 1.3 1999/12/31 08:36:21 stevemw Exp $`

6.5.1 What kinds of problems are related to X11 window managers?

Since AWT is (for better or worse) based on Motif, a Motif-like window-manager may be less likely to have problems.

In general, window managers are a problem for UNIX users because each has a varying degree of ICCCM (Inter-Client Communication Conventions Manual) standards compliance. Moreover, window managers come with a range of features and bugs.

Please to contribute what you know about your window manager and running Blackdown graphical Java to the FAQ.

6.5.2 Where is the ICCCM Standard?

See Christophe Tronche's conversion of the 2.0 version of this document by David Rosenthal at *http://tronche.com/gui/x/icccm/*.

One discussion of the issues is found at the GNOME developer site: *http://developer.gnome.org/doc/standards/wm/book1.html*.

6.6 Performance and Blackdown Java

This section discusses various performance issues with Blackdown Java, which has been one of the more active threads on the mailing list. Performance has traditionally been limited by CPU speed, memory, and Linux's one-to-one kernel thread model.

Last update:

`$Id: performance.sgml,v 1.2 2000/01/06 11:35:56 stevemw Exp $`

6.6.1 How well does Blackdown Java perform?

Stay tuned for more. Meanwhile, these links should be useful:

- The Volano Mark. *http://www.volano.com/benchmarks.html*.
- A Comparison of JVMs on Linux by J. P. Lewis of Dream Quest Images: *http://www.idiom.com/˜zilla/Computer/javaCbenchmark.html*.
- SPEC JVM98. Several JVMs are detailed at SPEC's test-suite page here: *http://www.spec.org/osg/jvm98/*.

6.6.2 How can I find out how much memory Java is using?

Nathan Meyers <nmeyers@javalinux.net> suggests this approach:

> My recommendation is to stop trying to make sense of the output from top and ps, and go to an unambiguous source. Take a look at the contents of /proc/<pid>/status, where <pid> is the process ID. This will, at any instant you examine it, give you (among other things) total current VM usage and RSS.

7 Other Questions Related to Java and Linux

This section covers random questions that didn't fit into the other categories.

7.1 General Questions about Linux

While the majority of this FAQ tries to focus on Blackdown Java itself, we attempt to answer a few Linux-specific questions here. There are many better places to learn about Linux than this FAQ, however.

Last update:

`$Id: general-linux.sgml,v 1.3 2000/01/07 05:47:57 stevemw Exp $`

7.1.1 Where can I find Linux FAQs, mailing lists, and documentation?

Here's a list of good places to start:

- The Linuxdoc project: *http://www.linuxdoc.org/*.
- The main Linux website: *http://www.linux.org/*.
- Linux kernels and libraries: *http://www.kernel.org/*.

7.1.2 What is glibc, and why does it matter?

The UNIX "libc" is the core set of functions to which most utility and application code links. You can see references to libc in the **intro(3)** manual page. For your information, functions described in section two (see **intro(2)**) of the manual are all inside the kernel, whereas section three describes those which may be standard, but are stored in separate libraries. To view these pages, make sure your core manual pages are installed, and type `man 3 intro`, for example.

In order to save disk space and memory, modern operating systems like Linux have taken to using "shared" libraries that don't move their contents into code which links to them during the build process. Although there is a `/usr/lib/libc.a`, most production code is linked to `/usr/lib/libc.so`, which is typically a link-editor script for selecting the actual libc.so (shared object) archive.

In short, lots of programs on your system share code out of one or more files as they execute. To see what shared code an executable needs, there is an optional Linux utility named **ldd(1)**. If it's on your system, you can type `ldd PROGRAM` to see its shared libraries.

Here's an example:

```
$ ldd /bin/ls
        libc.so.6 => /lib/libc.so.6 (0x40003000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00000000)
```

The older (and still most common) Linux libc was known as "libc5." The GNU/Linux development community sought to improve on it by adding thread-safety where possible, 64-bit support, better POSIX and XPG4.2 compliance, superior organization of the library sources, and multi-byte character locale support. These and other changes will bring even more robustness and quality to the Linux development environment. Although the changes are inconvenient to us sometimes, the end result should be worth it.

The new library is now called "glibc" because it's a library common across Linux and HURD, and enjoys a broad base of GNU developer contributions. (I don't know how much of the earlier Linux libc software was derived from GNU libc, which has been around for quite a while on its own, I think.)

For more information on GNU libc, see See *http://www.gnu.org/software/libc/libc.html*.

Since C is central to the history and development of UNIX and Linux, you might enjoy the links found on *http://www.lysator.liu.se/c/* as well.

7.2 General Questions about Java

This section contains replies to a selected list of general questions about Java.

Last update:

`$Id: general-java.sgml,v 1.12 2000/01/07 05:47:56 stevemw Exp $`

7.2.1 What is Java?

Java is an imperative, object-oriented language from Sun Microsystems with a syntax similar to C. Because it supports classes, it is sometimes compared to the C++ language. Java is also compared to Smalltalk in part because they are both interpreted and require all user-defined types to be objects.

Java was first developed for handheld devices in the early 1990s. After indicating that he wanted move on out of frustration with the company's unfocused technical direction, Patrick Naughton changed his mind when CEO Scott McNealy offered to team him up with a variety of talented Sun people to create a new computing environment. They would have mixed results, but Java survived. For more on Java's dramatic origins, see section 7.2.2 (Why was the Java language created?).

One of Java's most attractive features is the common interface it tries to provide the programmer to a given host platform. Java classes are normally compiled into a format which will run on an abstract CPU known as the "Java Virtual Machine" (JVM). The execution of this "Java Byte Code" on the JVM is then simulated by a byte code interpreter running on the target system. However, JVMs have also been implemented in hardware Sun advertises as *Java Processors*. Berkeley's *CPU Info Center* website has more information about Java chips.

Some JVM implementations, in order to run faster, use a technique known as "Just-In Time compilation" (JIT). In a JIT, byte codes are translated to native machine code in a controlled fashion when they are first encountered, and then executed natively on a host processor. When the cost of compilation is low and the byte code is executed many times during Java program execution, JIT compilation can produce dramatic performance improvements. For portability, Java is normally not compiled to native code and executed directly by the host processor. (See section 3.4 (Just In Time Compilers for Blackdown Java) for questions about JITs available to this implementation.)

The "architecture-neutrality" of Java and its standard windowing environment (called AWT) allow for a fair degree of platform independence. Sun markets Java for its "write once, run anywhere" (or "WORA") capability, but those developing serious applications with Java know that one must still test on targeted platforms. For example, moving an application between SPARC/Solaris and i396/Linux may reveal surprising differences despite the similarities of both platforms. Furthermore, Internet browsers such as Internet Explorer and Netscape Navigator implement Java differently enough so that in practice, considerable development effort must ensue in order to provide consistency to the user.

Java attempts to be a language of choice for developing network-based applications. It is multi-threaded or "concurrent," and much of its class library is thread-safe. Thread support is intended in part to help Java handle networking tasks. One of the design goals for Java was that it could provide a safer, more secure execution environment for code in distributed computing applications running in conditions as hostile as the Internet.

The JVM can be configured such that only a safer subset of Java instructions can be executed. "Applets," which are intended for use in a web browser, use what is known as a sandbox. The safety of individual byte codes is checked just once, at class load time. This ability to use a "verifier" on the byte code can improve the performance of sandboxed code without sacrificing safety. A sandboxed environment also attempts to provide more security for networked applications by limiting access to certain resources host platform resources. Moreover, Java does not allow direct memory access except through its Native Interface API (JNI), which is restricted for applets.

Rather than burden the programmer with the task of managing memory, Java uses "garbage collection" (GC) to automatically reclaim storage which is no longer being used by an application. This works well in practice: memory "leaks" (where the system refuses to reuse a chunk of memory which the application has finished using) are rare. Allocation efficiency is high. Unfortunately, garbage collection can sometimes cause problems in interactive or real-time applications: many GC implementations do their work in large batches, which can lead to long pauses at unexpected times. Unfortunately, current techniques for avoiding this problem typically lead to less efficient GC.

Finally, Java's support for modularity is quite good. A simple but reasonably effective package facility has been used to create a powerful class library that most Java programs use heavily. Access control to packages and classes is provided by mechanisms which are fairly easy to use, yet quite effective.

For a more detailed look at the language, please see *The Java Language Environment: a White Paper* at *http://java.sun.com/docs/white/langenv/* by James Gosling and Henry McGilton in May 1996. You may want to go to a less flowery description of the Java language found in *The Java Language Specification* by James Gosling, Bill Joy, and Guy Steele at *http://www.javasoft.com/docs/books/jls/*. You can get an inside view of the JVM in *The Java Virtual Machine Specification* by Tim Lindholm and Frank Yellin at *http://www.javasoft.com/docs/books/vmspec/*. For questions about Java security, see the links in section 7.2.4 (Is Java secure?).

7.2.2 Why was the Java language created?

In January of 1991, Patrick Naughton, James Gosling, Mike Sheridan, and Sun founder Andy Bechtolsheim went to stay with Bill Joy in Aspen to brainstorm. Together, they felt that a dramatic change was needed in the direction Sun had been taking with hardware and software, and CEO Scott McNealy had given them permission to work on a secret project to develop a new product. They wanted to make a revolutionary wireless, hand-held platform. While there, they realized that they needed a programming environment to succeed. Bill had talked about a language environment using C++ which he had envisioned for a 64-bit OS called Spring at Sun Labs. Instead, James would take a different direction.

For 18 months, Ed Frank and Mike Sheridan led the resulting project known as "Green." Their team included Chris Warth, Craig Forrest, Jonathan Payne, Jim Graham, Wayne Rosing, Joe Palrang, and Chuck Clanton. Among the results of this frantic effort would include the Oak programming language, for which James takes credit. Given Bill Joy's interest in C++ for Spring, it's no accident that the two languages have several common features.

The remaining history of the Green project is interesting, but suffice it to say that Oak, which evolved into Java, survived when Green failed to achieve an immediate customer. Despite Green's mixed results, Java would become very important to Sun. Since Java's developers were already interested in integrating it into Web-based applications, Webrunner (the first Java web browser) inspired developers to create the first Java applet in early 1994.

Source: *The Java Handbook*, written by Patrick Naughton and published in 1996 by McGraw-Hill (ISBN 0-07-882199-1).

7.2.3 Where can I learn more about Java?

Here are a number of excellent starting points.

- Elliotte Rusty Harold's *comp.lang.java* FAQ: *http://sunsite.unc.edu/javafaq/javafaq.html*
- The Java networking FAQ: *http://www.davidreilly.com/java/java_network_programming/*
- Peter van der Linden's Java Programmer's FAQ: *http://www.best.com/~pvdl/javafaq.html*
- Javasoft's Developer Connection FAQ: *http://developer.javasoft.com/developer/techDocs/faqs.html*

7.2.4 Is Java secure?

It has features that make it a candidate for being more secure than other choices. To read more about Java security, see the page *Java Security FAQ: the Unofficial Answers from the Princeton Secure Internet Programming Team* at *http://www.cs.princeton.edu/sip/faq/java-faq.php3.*

7.3 CJK Issues with Blackdown Java

This section discusses problems with displaying and inputting Chinese, Japanese, and Korean language characters in Blackdown Java applications.

Last update:

```
$Id: cjk.sgml,v 1.3 1999/12/31 08:36:16 stevemw Exp $
```

7.3.1 Can I display and input Kanji with Blackdown Java?

Yes, but the information we have to offer on this is sketchy.

In addition to the information found here, you will certainly appreciate Dan Kegel's page, *Fonts and Internationalization in Java* found at *http://www.alumni.caltech.edu/~dank/javafont.htm.* You are encouraged to visit Javasoft's 1.1 discussion of this topic at *http://www.javasoft.com/products/jdk/1.1/docs/guide/intl/fontprop.html*, as well.

Please be aware that these FAQ entries are sparse. If you have additional information about making CJK (Chinese, Japanese, and Korean) code work well with Java on Linux, please tell the FAQ maintainer.

Library Issues The following entries also relate more to situations that arise when you must settle on one set of libraries or the other, and you end up without being able to run some application or another because of unresolved symbols. Keep in mind that with their own builtin loader and libc, recent versions of the JDK shouldn't cause you to change your libraries at all.

Libc5 Systems *Nozomi Matsumura <nmatsumu@can.bekkoame.ne.jp>* has a suggestion that at least applied to earlier, `libc5` versions of the JDK, and may still work. It involves making use of multi-byte character support provided in builds of `libX11.so` with the **Xsetlocale(3)** call.

Matsumura points us at this fix provided by *Hiroshi Hisamatsu* <*hisama2@asahi.email.ne.jp*>, which is a small, shared library (`liblocale.so`) to explicitly specify the Kanji locale. It is available from *http://www.asahi-net.or.jp/~nf6h-hsmt/jdk1.1.1-locale.tar.gz*.

More detail on this is available in Japanese at *http://www.asahi-net.or.jp/~nf6h-hsmt/java.html*. The fix also involves adding the following lines to your `.java_wrapper` file in `$JAVA_HOME/bin`:

```
LD_PRELOAD="$JAVA_HOME/lib/${ARCH}/${THREADS_TYPE}/liblocale.so"
export LD_PRELOAD
```

Matsumura also mentions changing the locale specified by your environment variable `LANG` to `ja`, and says that `ja_JP.{EUC,ujis}` did not work.

Libc6 (glibc) Systems In addition to the properties hacks mentioned above (see section 7.3.1 (Libc5 Systems)), the inverse of what `liblocale.so` does is probably possible on glibc-based systems, if there's a problem (with some other application) at all. You may be able to simulate **Xsetlocale(3)** by wrapping **setlocale(3)** with it. For example, libc5 versions of XFree86 (see *http://www.xfree86.org*) handle locale with the **Xsetlocale(3)** call, but that changes under glibc to plain **setlocale(3)**. See the **setlocale(3)** manual page for more information.

Can I enter two byte Kanji with XIM? On July 14th, 1998, *Nozomi Matsumura* <*nmatsumu@can.bekkoame.ne.jp*> also submitted this information acquired from *Minori Takagaki* <*lennen@st.rim.or.jp*>, who was able to input and display Kanji using TextComponent.

Takagaki was able to input directly using kinput2(XIM) or by cut&paste in Java applications and in appletviewer.

Takagaki's environment is:

- Slackware-3.4 + PJE-0.1(*1)
- Kernel: 2.0.34
- libc: 5.4.38
- jdk1.1.6-v1-libc5

Takagaki said to use Japanese characters, you need to setup as described here:

1. Install Red Hat Motif 2.1 and X-TT(*2)
2. Modify `font.properties.ja` (to use X-TT fonts)
3. Get `liblocale.so` (see the discussion of this above in 7.3.1 (Libc5 Systems))
4. modify `.java_wrapper`

```
export LD_PRELOAD=liblocale.so
export LANG=ja
export DYN_JAVA=1 # to use libXm.so.2 from RH Motif2.1
```

5. If using appletviewer, modify `$JAVA_HOME/bin/i686/green_threads/appletviewer` to use the dynamically linked java binary.

X-TT is free a X font server for using Japanese TrueType Fonts. See *http://www.ryukyu.ad.jp/~shin/linux/x-tt/*.

But some problems occur:

1. The `getText()` function is O.K. but `setText()` is **not**.
2. Mouse drag selection is **not** well. Strings with Japanese characters using TextField/TextArea are **not** selected correctly.
3. When a cursor moves in TextField, Japanese characters are often modified randomly.
4. When Japanese characters are input in TextAreas, display timing is very strange. But `getText()` works correctly. Example: to input 'NIHONNGO abcdef' when keying 'ab', 'NI' shows; when keying 'cd', 'HONN' shows, etc. . .

Editor's note: see section 6.4.3 (TrueType Support in X11).

7.4 Blackdown Java Y2K Issues

Questions about Blackdown's Y2K readiness or lack of it.

Last update:

```
$Id: y2k.sgml,v 1.2 1999/12/30 12:01:41 stevemw Exp $
```

7.4.1 Are there any Y2K problems with Blackdown Java?

Blackdown Java hasn't been y2k-tested, but according to *Scott Hutinger <srh@www.mprojects.wiu.edu>*, the current JDK 1.2.2 RC included some date patches from one of our Sun Liasons. For now, please see *http://java.sun.com/features/1999/08/y2k.html* for more information.

You're welcome to contribute what you know to the FAQ.

8 Blackdown Java Development and Platform Porting

This section answers a few questions about development and Blackdown Java. First, it discusses using Blackdown Java to develop applications. Then it addresses the development of Blackdown ports themselves. It also describes how you can take advantage of the work the Blackdown team is doing, and how you can contribute yourself.

8.1 Development with Blackdown Java

This far from comprehensive section includes discussion of full-blown development with Blackdown Java in the Linux environment.

Last update:

`$Id: java-development.sgml,v 1.3 1999/12/31 08:36:18 stevemw Exp $`

8.1.1 How can I learn more about developing Java on Linux?

One of the best ways to get started is by participating on the Java Linux mailing list. If you have access to Linux, you can at least experiment with Java.

Nathan Meyers <nmeyers@javalinux.net> has written *Java Programming for Linux*, published by Macmillan Computer Publishing; its ISBN number is 1571691669. See *http://www.javalinux.net/* for more information.

8.1.2 How do I compile native C or C++ code for interfaces?

See section 4.5.3 (Compiling Library Code for JNI).

8.2 Java Code Examples

This section has pointers to a few code examples we're adding in support of this document. Feel free to contribute suggestions.

Last update:

`$Id: examples.sgml,v 1.2 2000/01/11 06:50:21 stevemw Exp $`

8.2.1 Where can I find some example Java code to get started?

There isn't much documentation yet, nor are there many examples, but please take a look in this directory: *http://www.blackdown.org/java-linux/docs/faq/examples/*.

8.3 Java Licensing and the Open Source Community

A discussion of licensing issues with respect to developing Java on Linux. This section also addressess some of the questions those who are interested in contributing to JVM development may have with respect to licensing. None of this is legal advice!

Last update:

`$Id: licensing.sgml,v 1.15 2000/01/24 08:36:39 stevemw Exp $`

8.3.1 What issues are relevant to understanding Java licensing?

As a user, you're subject to the same licensing restrictions with Blackdown Java as you would be with Sun's Java binaries on Solaris or Win32. It's as simple as that. Pointers to the relevant Sun licensing `README` files are given below. In other words, you may copy Blackdown JDK binaries over the network from Blackdown mirrors.

As a developer, you unfortunately may *not* redistribute the Blackdown JDK on CDROM without permission from Sun. Sun has suddenly become very clear about this. Commercial Linux distribution vendors such as Red Hat must work with Sun directly to secure such rights.

The JRE (Java runtime environment) may be embedded and redistributed in your application.

J2SE (Java 2 "standard edition") source is available strictly through Sun to those willing to agree to a license containing certain intellectual property restrictions. Your strict source differences ("patches") may be exchanged freely and openly so long as they *only* address the porting of the code from one platform to another.

Blackdown is required to test its "full releases" of Java against a Sun-supplied test suite (see section 8.5 (Java Compatibility Kit)). If they don't pass, then they must be clearly labeled as "release candidates."

(This topic is briefly introduced in section 2.3.4 (Under what license is Blackdown Java available?).)

Links to Java Binary Licenses Below are links the the various license-related documents Blackdown has distributed on behalf of Sun with recent builds of each platform:

- **JDK 1.02** *http://www.blackdown.org/java-linux/docs/faq/README/1.0/COPYRIGHT*, Known as the *JDK BCL 9-6-96*, or *Java Developer's Kit Version 1.0.2 Binary Code License.*
- **JDK 1.1.8** *http://www.blackdown.org/java-linux/docs/faq/README/1.1/LICENSE*, or simply the *Binary Code License Agreement.*
- **JDK 1.2.2** *http://www.blackdown.org/java-linux/docs/faq/README/1.2/LICENSE*, likewise called the *Binary Code License Agreement.*

Licensing for Java Source Community access to Java product source has had an inconsistent history. Sun is pursuing a new path with important aspects of its intellectual property licensing known as "Community Source." This is a clear compromise between simply giving away source code (the files from which a set of executables like the Java platform is built) and never sharing it with the world.

This document can't explain everything you may want to know about licensing and development, but it can give a synopsis of the four most important issues to you as a developer or interested user.

- **SCSL, or Sun Community source License.** Sun is distributing source for many (but not all) of its Java products, including the core JVM. Selling products based on changes you make to the code is something you'll have to work out with Sun with branding and distribution licensing. Blackdown works closely with Sun to ensure that it stays within guidelines, among which include the restriction that it can only distribute patches (diff'erences). You must get the code from Sun directly.
- **Burgeoning Java APIs.** Although the core of the Java product has been available in source form to the Blackdown community of software developers from early on in its history, the ever-growing number of Java APIs often take longer to go into source availability. Some may never go into community source release.
- **NDAs.** Blackdown developers have historically needed to sign non-disclosure agreements with Sun, and are now forbidden to participate directly in the two most-commonly known cleanroom open-source JVM efforts, Japhar (*http://www.japhar.org/*) and Kaffe (*http://www.kaffe.org/*).
- **JCK (Java Compatibility Kit).** In order to release a build of Java and call it "ready for production," one must test it thoroughly with Sun's Java Compatibility Toolkit, or JCK. Because this suite is closely knit with Sun's partner development process, and is presumably a part of its current revenue stream plans for Java, obtaining access to has been difficult. Read more about the JCK in section 8.5 (Java

Compatibility Toolkit). *Blackdown has found this toolkit to be essential for porting JDK 1.2 partly because of its increased complexity.*

8.3.2 Licensing Comparisons

- **SCSL**. Read more about the changing horizon of Sun's engagement with the open source community at *http://www.sun.com/software/communitysource/*.
- **LGPL,** what Japhar uses. See *http://www.fsf.org/copyleft/lesser.html* for information about the "GNU Lesser General Public License."
- **GPL,** and Kaffe's perspective. See *http://www.kaffe.org/FAQ.html* for information.
- Other such licenses are available at: *http://www.opensource.org/licenses/*.

8.3.3 Blackdown's Position on Sun's Community Source Policies

Blackdown developers have worked with Sun through each major version of the Java product family, and have appreciated its historic willingness to share source with certain interested developers. We hope Sun's recent attempt to formalize a community source process will prove successful for J2SE (Java 2, standard edition), Sun's commercial partners, and communities like Blackdown. After years of effort to secure a formal code codevelopment relationship, Blackdown is now ready to help Sun and Inprise reach out a world of developers wanting to contribute to the Java platform. However, there remain issues that Blackdown has always struggled with Sun to resolve:

- *Help for developing a consistent policy for recruiting new Blackdown developers.* Currently, we are unclear how to add new developers to the CVS repository access list, an essential step in bringing in new people. With all the new code to port and maintain, Sun surely knows this is critical.
- *Java compatibility kit (JCK) access and updates.* The JCK is essential for port development, yet Sun has been reluctant to share it consistently with all Blackdown developers.
- *Access to highly current defect information.* Blackdown is never sure when it is fighting a defect that Sun has already solved in a different way.
- *A standardized method of obtaining recent product changes from Sun and its partners.* A shared CVS repository between Sun and Blackdown would be ideal, but daily differences could also be exchanged.
- *Standardized codevelopment procedures.* In general, a commitment on Sun's part to build standard policies with respect to community source concerns rather that falling back on the heroics of individual managers like Steve Byrne and Calvin Austin.

If Sun could address these issues, we believe it would usher in a new era of productivity among open source developers wanting to help Java succeed.

Please see section 1.2.1 (Statement of Purpose for the Blackdown Project) for a formalized explanation of Blackdown's philosophy about participation with the commercial software world.

8.3.4 Where can I find Sun's Community Source Agreement?

For the most recent set of instructions, which apparently allow you to obtain the source for J2SE (Java 2 Standard Edition), see *http://www.sun.com/software/communitysource/java2/*.

In the past, you first you had to read, sign, and (historically) FAX in an agreement with Sun. The agreement could be found at *http://www.javasoft.com/nav/business/source_form.html*.

8.4 Compiling Blackdown Java from Source

Here are some answers to questions about Java platform source code, compiling, and contributing fixes.

Last update:

```
$Id: source.sgml,v 1.8 2000/01/11 06:50:23 stevemw Exp $
```

8.4.1 Can I build Java myself?

Building the JDK on Linux with patches supplied by the Blackdown project is fairly straightforward. We encourage you to read *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux.src* from the patched source distribution if you plan to undertake this project.

8.4.2 Could you give me a pointer to Sun's licensing agreements?

Please see section 8.3.4 (Where can I find Sun's Community Source Agreement?).

8.4.3 How do I build Blackdown JDK 1.2 now that I have the source?

Read Juergen Kreileder's instructions in your binary distribution. It can be found online here: *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux.src*.

8.4.4 What about Motif? Don't I need that?

You can build the JDK against Lesstif, the *Hungry Programmers'* GNU copylefted implementation of the Motif API. For more information, visit their website at *http://www.lesstif.org*. Make sure you get the "current" distribution. They are very responsive to accurate bug and problem reports.

On the other hand, a number of Motif developer library vendors exist for Linux systems. I recommend you read the Motif *comp.windows.x.motif* news group FAQ. You can find it here: *http://www.rahul.net/kenton/faqs/mfaq_index.html*.

8.4.5 What version of Motif is the Blackdown team using?

Thanks to Jim Burmeister at Metro Link, the Blackdown team is currently using metromotif-2.1.10-4. See *http://www.metrolink.com/motif/index.html* for more information.

8.4.6 When I build and install 1.1, I get unsatisfied link errors at runtime!

For JDK 1.1.x, make sure you're building all the targets Steve Byrne suggests in *http://www.blackdown.org/java-linux/docs/faq/README/1.1/README.linux.src*. Here's an example:

```
make DESTDIR="$ALT_JAVAHOME/jdk116v2b" RELEASE="MY-JDK1.1.6-V2B" \
        appletviewer jre_only 2>&1 | tee MakeOut-V2B1-BUILD-01
```

In this example, I have `ALT_JAVAHOME` set to the `javasrc/build`, just above the `genunix/` target directory. You should be able to set `ALTMOTIF_DIR` to where you've installed Lesstif.

Note: you ought to have `ALT_BOOTDIR` set to a running JDK 1.1, and I build with `CLASSPATH` and `JAVA_HOME` unset.

8.4.7 Can I try builds of the latest JDK port in development?

Please stay tuned to the mailing list for announcements.

8.4.8 What about the JCK?

See section 8.5 (What does Blackdown use to test Java's adherence to Sun's standards?) for more information.

8.4.9 How can I contribute to Blackdown Java development efforts?

If you're not already contributing good bug-reports, patches, or documentation, those are places to get started. If you want to join the developer team, please indicate your interests, relevant experience, and a couple of professional references to *Karl Asha <karl@blackdown.org>*, the list administrator.

In addition to programming in C, Java, Motif, X11R6, using GDB, and the occasional kernel diving, you'll need to understand CVS, the GNU concurrent versioning system maintained by Cyclic Software *http://www.cyclic.com/*. A good place to learn about it is Karl Fogel's *Open Source Development with CVS* book published by Coriolis; its ISBN number is 1-57610-490-7, and you can find much of it freely online at *http://cvsbook.red-bean.com/*.

8.4.10 What is a source patch?

A source "patch" is generated by the **diff(1)** program by comparing the contents of one file (or sets of files, including whole directory trees) with another and producing a set of differences suitible for the **patch(1)** utility. When Blackdown has released a set of patches for a given Sun source release, you can build what Blackdown is shipping on your own Linux system by applying the patch set to your Sun source base. Due to time constraints and the occasional issue with Sun, Blackdown can't always issue diffs with every binary release. However, it is one of Blackdown's goals to deliver source differences as often as possible.

For more *http://www.gnu.org/software/libc/libc.html*.

8.5 Java Compatibility Kit

Discussion of Sun's JCK, a Java platform testing suite.

Last update:

```
$Id: jck.sgml,v 1.7 2000/01/05 18:35:31 srh Exp $
```

8.5.1 What does Blackdown use to test Java's adherence to Sun's standards?

With the advent of Java 2.0, Blackdown is using the JCK, a large suite of tests including a harness. Before Blackdown can stop calling a build a "release candidate" and call it "Java," it must pass the test. This test suite is sometimes referred to as the *Java Conformance Toolkit*, but glossary calls it the *Java Compatibility Kit*.

Currently, sun's JIT code is one of the areas that requires careful scrutiny during a release.

Blackdown continues to work with Sun to obtain the latest copies of this code, but it requires tremendous effort to stay synchronized with new tests continually pouring into Sun's internal version.

8.5.2 Does any build of Blackdown Java 2 pass the JCK?

For the latest, please visit *http://www.blackdown.org/java-linux/jdk1.2-status/jck-status.html*.

8.5.3 Will Sun make the JCK more accessible?

Blackdown developers are working with Sun and Inprise to resolve the issues surrounding community source for Java products such as the JCK. We assume Sun will deal with this on a case by case basis. At this time, Sun developers working on the Jini connection API (a distributed computing platform) have had stronger ties to the open source movement than most other APIs. See *http://www.jini.org/* for more detail.

8.5.4 Is there any online documentation for the JCK?

See section 8.5.5 (The Latest Blackdown JCK Notes) below.

The FAQ needs more answers about the JCK, but for now, here are some links:

- Carla Schroer's 1998 Java One presentation: *http://java.sun.com/javaone/javaone98/sessions/T203/*
- A chat session about the test kit: *http://developer.java.sun.com/developer/community/chat/JavaLive/1997/jl0218.html*
- Carla Schroer's 1997 Java One presentation: *http://java.sun.com/javaone/sessions/slides/TT28/startit.html*; a transcript of her talk is provided by the Irish Java User's group: *http://sunsite.compapp.dcu.ie/IJUG/javaone/transcripts/compatibility.html*;

8.5.5 The Latest Blackdown JCK Notes

Scott Hutinger <srh@www.mprojects.wiu.edu> has been working with the JCK and has contributed this information on December 29th, 1999 in hopes that it will be helpful for new users getting started with the JCK:

- Always make sure `classes.zip` is unzipped.
- Always make sure (when using slave) that `classes` and `javatest.jar` are in the `$CLASSPATH` correctly when using the map file and `-mapArg`. Note: the mapped files is only needed when doing slave/master via NFS.
- Only about 20 failures from these two items not being in the class path for compiler tests.
- *Fonts and the JCK documentation* Note that '-' are removed; on most releases, i.e. `-DDISPLAY` is `DDISPLAY=`, or `-mapArgs = mapArgs` due to font issues.
- Don't use `file.txt` (`.txt`) for the ftp file:
  1. Set in the file: `lib/javasoft-x.jte`.
  2. Set `ftpURL=ftp://someplace/somefile.txt`
  3. This will cause a failure, use `//someplace/someothertype` (i.e. `.gif`).
  4. Use something like `.gif`, as `.txt` will be plain/text, and the JCK thinks it should be html/text or something like that.
- Bad setup for java-compiler (solaris), use `share ..... root=myclientslave.edu .....` unless the Solaris machine is setup well, the slave needs root access. This is mainly for `*-compiler-*` tests. As the gui will show unexpected failure from the compiler (bad access rights), although it's not stated.
- Native threads took around 23 hours to run the java-compiler tests; Green threads on the slave took about 8 hours?!
- The intial tests are not in the `compiler.jtp` file, although if failures exist, just stick in the correct `javasoft.sqe.javatest.selection.initialFiles=` setting.
- At that point the complete tests don't need to be run, just the failure portions to narrow down what needs to be fixed or changed.
- JNI and the JCK (`JCK-runtime-vm-x`) build in `tests/vm/jni/build`.
- Using `ssh` to `ssh localhost` for getting the display set correctly will make some of the URL type tests fail.

9 History and People of the Blackdown Java Effort

About the people who have contributed to the Blackdown Java porting project.

9.1 History of the Blackdown Java Effort

This section is about Blackdown's past. In order to understand the daily hype about Java (including what you hear about Blackdown), it's important to look back over its history. If you have anything to add to this section, please contribute.

Last update:

```
$Id: past.sgml,v 1.3 1999/12/31 10:50:58 stevemw Exp $
```

9.1.1 What is the history of the Blackdown project?

Until we can expand this section, here is just one tidbit to offer; the bug that popularized the request for Java on Linux at Javasoft: *http://developer.java.sun.com/developer/bugParade/bugs/4097810.html*. Note that the Blackdown effort existed (and was arguably as important to the Linux community then as now) well before *Bug Parade* did!

There were a number of complex issues involved with this request, including the wide variety of Linux distributions, target architectures (beyond Intel where Javasoft already supports Solaris x86), the huge number of announced APIs with required native support, and so forth.

However, Javasoft finally released the JDK 1.2 sources to the Blackdown team, and provided measured engineering resources to help complete the port. You can read more about this in a press-release published on Javasoft's website. See *http://www.javasoft.com/pr/1998/11/pr981102-01.html*. For the information about Sun's eventual decision to support Java on Linux, see 2.3.5 (Is Javasoft going to support Java on Linux?).

For more information about Java's history, see section 7.2.2 (Why was the Java language created?).

9.1.2 Other Historical Information

Please see the far from complete biographical information in section 9.2 (Who Ported the Blackdown JDK?).

9.2 About the People Behind Blackdown Java

Any software project is about people. Here are some of the highlights. We've left out a whole list of contributors; please see the various distribution README files for details.

Last update:

`$Id: people.sgml,v 1.6 2000/01/11 06:50:22 stevemw Exp $`

This section answers questions about the people who've contributed to the Blackdown project.

9.2.1 Who Ported the Blackdown JDK?

The Java 1.02 Port: *Randy Chapman <chapman@wilbursoft.com>* did the original port of the Solaris JDK 1.0 to Linux on the i386.

The Java 1.1 Ports: Steve Byrne did much of the work on the Intel port of Java 1.1.

Quoting Steve Byrne from *http://www.blackdown.org/java-linux/docs/faq/README/1.1/README.linux*

> Special thanks to Johan Vos (Sparc Linux port), Chris Seawood (GLIBC), Karl Asha (who gave us blackdown, thanks Karl!) Kevin Buettner, Scott Hutinger, Kevin Hendricks, Brian Keller, Philippe Laliberte and the rest of the PowerPC porting team – these guys really fixed a *lot* of generic JDK bugs – the 1.1.5 port is *much* better as a result of their hard work. Dan Powell and Rich Cannings have been very gracious to take over the administrative duties and to provide a site (and domain name) dedicated to java-linux – thanks! [Also: Neal Sanche fixed the thread creep bug. Thanks, Neal!]

The Java 1.2 Ports: Regarding work done on Java 1.2, the *http://www.blackdown.org/java-linux/docs/faq/README/1.2/README.linux*, file indicates:

> The following people have helped with the porting of Java 2 to Linux:
>
> Juergen Kreileder, Kevin Hendricks, Steve Byrne, Johan Vos, Anand Palaniswamy, Eddie Dost, David Connelly, Karl Asha, Kars de Jong, Scott Hutinger, Calvin Austin, Nigel Gamble, Michael Sinz and Brad Crochet.

Collecting History If we've missed anyone, please let us know. Please pass on historical information to the FAQ maintainer.

9.2.2 Who is Steve Byrne?

Steve Byrne was educated at Carnegie-Mellon University, and among other things, managed Java-related projects at Sun Microsystems. In addition to contributing much of the Blackdown JDK 1.1 port to Linux, he has also been a long-time GNU SmallTalk maintainer. Steve has a small amount of info about himself at his web page: *http://www.penguinis.org/˜sbb/*. For more information about GNU SmallTalk, see the *Official GNU Smalltalk Page* at *http://www.smalltalk.org/versions/GNUSmalltalk.html*.

9.2.3 Who is Karl Asha?

Karl Asha from the beginning has been the maintainer of the web site, Blackdown's Java-related mailing lists keeper, and has managed product distribution. He's now taking on the central role of licensing administration, which can be a sticky wicket.

9.2.4 Who is Juergen Kreileder?

His first name spelled more correctly would be "Jürgen." He hangs out at University of Dortmund Fachbereich Informatik, and hacks on everything Java and Linux. Juergen has taken on an increasing number of technical management roles on the Blackdown Java project.

9.3 Information on the Java-Linux FAQ

This section gives some background information about the FAQ itself.

> One of the amusements of idleness is reading without the fatigue of close attention; and the world therefore swarms with writers whose wish is not to be studied, but to be read. – Samuel Johnson, November 11, 1758.

From *Writing: the Samuel Johnson Sound Bite Page* quoted in number 948 at *http://www.samueljohnson.com/writing.html*.

Last update:

```
$Id: about-faq.sgml,v 1.13 2000/01/11 07:04:38 stevemw Exp $
```

9.3.1 Who maintains the FAQ?

With help from the Blackdown Development Team, Stephen Wynne is the current FAQ maintainer. However, answers to questions found in this document are from a variety of participants on *java-linux@java.blackdown.org*. If you have any additions, comments, clarifications, or corrections, please send them to *Stephen Wynne <stevemw@place.org>*.

9.3.2 Where can I get a copy of the FAQ?

The most recent copy of this FAQ can be found at the *Blackdown Documentation Page*: *http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux.html*.

- Here's the text version of this document:
  http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux.txt.
- The PostScript "letter" version:
  http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux.ps.
- The PostScript "A4" version:
  http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux-a4.ps.
- The PostScript "letter" version created with **enscript(1)** from the text above and rendered into two columns:
  http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux-r.ps.
- The PDF "letter" version created with **pdflatex(1)**:
  http://www.blackdown.org/java-linux/docs/faq/FAQ-java-linux.pdf.

Full SGML source for the FAQ is available in the same directory: *http://www.blackdown.org/java-linux/docs/faq/*.

9.3.3 Please help the FAQ maintainer!

The FAQ maintainer is looking for the following information:

- Corrections of all kinds, especially with this larger version.
- Linux distribution-specific information. What did you have to do to get the latest version of 1.1 working? The latest 1.2 version?
- Native code integration you want to get credit for contributing.
- How you got a certain Java API to work.
- Performance tricks for Java 2.
- Your Java Compatibility Kit (JCK) experiences.
- Any wide-character (CJK) input/display setup tips and suggestions you may have, especially for Java 2.
- Kernel issues and Java. (Such as threads and scaling, 2.3 kernels.)

9.3.4 Are there any plans for automating the FAQ?

Not at this point. The FAQ requires a coherent effort to maintain, and dividing it among a relatively anonymous group of people might not help. One step we've recently taken is to move the source for it into CVS, and Blackdown developers are now beginning to share in its maintenance.

9.3.5 Caveat Emptor

This document may contain technical information that is incorrect, out-dated, and may *unintentionally* misrepresent the opinions of *java-linux@java.blackdown.org* participants. Neither the FAQ maintainer nor the members of the mailing list make any warranty with respect to its usefulness or safety.

9.3.6 Terms of Use

9.3.7 Major Contributors to this Edition of the FAQ

Although the FAQ maintainer takes responsibility for any errors in this document, some people have made a tremendous difference in producing this edition. Their patient assistance has been greatly appreciated.

- *Barton C Massey* <*bart@cs.pdx.edu*> helped with suggestions and technical review for section 7.2.1 (What is Java?).
- *Scott Hutinger* <*srh@www.mprojects.wiu.edu*> contributed JCK and Arm information, as well as overall review of the FAQ that included corrections and his own improvements.
- *Juergen Kreileder* <*kreilede@issan.cs.uni-dortmund.de*> wrote much of the new JDK 1.2.x documentation referenced here, and has been helpful in documenting TrueType problems.
- *Karl Asha* <*karl@blackdown.org*> provided general review.

9.3.8 Other Acknowledgements

This FAQ's content and structure was inspired by several documents produced by Eric S. Raymond. Eric also maintained an early HOWTO regarding Java on Linux. It's archived at *http://www.tuxedo.org/˜esr/faqs/java.html*.

We've archived one of the text versions of this document (in this case one Karl Asha wrote) plus a `MINI-HOWTO` of uknown provision:

http://www.blackdown.org/java-linux/docs/faq/README/1.0/Java-Linux-HOWTO

http://www.blackdown.org/java-linux/docs/faq/README/1.0/Java-Linux-MINI-HOWTO

The FAQ was produced directly from SGML with the SGML tools package available from *http://www.sgmltools.org* v1.0.9, using the `LINUXDOC.DTD` (both available for in RedHat 6.1 in standard RPMs).