## Java for the Linux platform

# COFFEE WITH MILK AND SUGAR

**Anyone who has worked with Java programs will be aware of their great advantage: platform independence. Java programs do however need a runtime environment for execution. In this article, Sebastian Eschweiler tells you what options are currently available under Linux**

**Java SDKs or JDKs** Java Software Development Kits provide the entire Java environment for the programmer. You will only need an SDK if you want to develop Java programs yourself. If there is a JDK in place, you need no separate JRE in order to execute programs – the JDK already includes all functions of the JRE.

**JRE** Java Runtime Environment. Since Java programs only exist in the so-called byte code (a sort of Java machine code), a Java interpreter must be used for execution, which can be found in the JRE.

While Microsoft is trying to ban Java completely from the latest version of Windows (Windows XP) due to the licence dispute with Sun, the Linux user has more options than ever to make his system Java-capable. This article gives you a short overview of the existing **Java SDKs** and **JREs** available for Linux.

## Spoilt for choice

In addition to the known JDK/JRE from Sun, Java users have a huge choice of alternatives under Linux. Here are the main Java Kits for the Linux platform:

- Sun JDK 1.3.1/1.4 Beta 3
- Blackdown 1.3.1 FCS
- IBM Java 2 SDK/JRE 1.3
- Kaffe 1.0.6

Let's take a look first at the best-known option for making a Linux system Java-capable: the Sun-JDK/JRE.

## Java from the inventor

At present, Sun is in between the two JDK versions 1.3.1 and 1.4. Version 1.4 is in fact still in Beta status, but is nevertheless already highly stable and highly recommended for private use. The Sun-JDK or JRE can be found at *java.sun.com*.

Whether you now settle on the JDK or the JRE, it makes little difference to the installation procedure. But we describe below only the JDK installation. When downloading from the aforementioned Web site you can choose between a *tar.gz* archive and an *rpm* package. Depending on which format you choose, after download one of the two following files should be on your hard drive:

```
* j2sdk-1_4_0-beta2-linux-i386-rpm.bin
* j2sdk-1_4_0-beta2-linux-i386.bin
```

The respective *bin* file must first be made executable:

```
chmod a+x j2sdk-1_4_0-beta2-linux-i386-rpm.bin
```

or

```
chmod a+x j2sdk-1_4_0-beta2-linux-i386.bin
```

Now the program can be started:

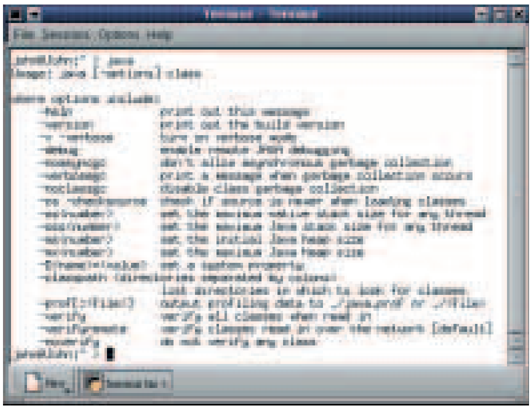
```
./j2sdk-1_4_0-beta2-linux-i386.bin
```

You will first be confronted by the licence agreement. After confirmation, the JDK is installed in a sub-directory *j2sdk1.4.0* of the current directory. When using the archive with *rpm in the name, after confirming the licence agreement, the rpm* file will be made, which can then be installed by the *root* administrator with the command

```
rpm -iv j2sdk-1_4_0-beta2-linux-i386.rpm
```

Once installation is complete you must still set two

Figure 1: Java Web site from Sun (java.sun.com)



```
[illegible]:~ > java
Usage: java [-options] class

where options include:
    -help             print out this message
    -version          print out the build version
    -v -verbose       turn on verbose mode
    -debug            enable remote JAVA debugging
    -noasyncgc        don't allow asynchronous garbage collection
    -verbosegc        print a message when garbage collection occurs
    -noclassgc        disable class garbage collection
    -cs -checksource  check if source is newer when loading classes
    -ss<number>       set the maximum native stack size for any thread
    -oss<number>      set the maximum Java stack size for any thread
    -ms<number>       set the initial Java heap size
    -mx<number>       set the maximum Java heap size
    -D<name>=<value>  set a system property
    -classpath <directories separated by colons>
                      list directories in which to look for classes
    -prof[:<file>]    output profiling data to ./java.prof or ./<file>
    -verify           verify all classes when read in
    -verifyremote     verify classes read in over the network [default]
    -noverify         do not verify any class
[illegible]:~ >
```

Figure 2: Java commandline options

important system variables, in order that the now-installed Java environment can be found by Java application programs:

```
export JAVA_HOME=/usr/lib/j2sdk1.4.0
export PATH=$PATH:/usr/lib/j2sdk1.4.0/bin
```

## Blackdown

The developer group at *http://www.blackdown.org* also provides a JDK or JRE. The objective of this software group is to port Java on the basis of the Sun source code onto Linux. You will now be wondering why they bother, when there is already a Linux version from Sun available. Blackdown promises to bring in special adaptations for the Linux platform, as the result of which the program package should be more stable and faster. Since people are working on the basics of the Sun implementations, it will therefore take some time before the corresponding version numbers are attained by Blackdown. The latest version at present is 1.3.1.

Opinions vary widely on the question of whether to rely on a version from Sun or whether the Java implementation from Blackdown should be used. But it has already been shown on many occasions that implementations from Blackdown work very reliably. So it's well worth taking a look at this alternative.

Installation is again very simple and is finished in a few steps. Whether you now decide on the JDK or merely want to use the JRE, this will not affect the installation in any way (apart from the directory names, obviously). Once you have downloaded the **tar.gz** archive from the Web site, there follows the usual procedure for extracting the archive:

```
tar xjvf j2sdk-1.3.1-FCS-linux-i386.tar.bz2
```

You will then find a new directory *j2sdk1.3.1* in the working directory. And with Blackdown, too, you must not forget to set or to adjust the two system variables PATH and JAVA_HOME as described above.

## Big blue

In recent times, IBM too has recognised the importance of Java and is offering JDK and JRE. The available versions can be found at *ibm.com*. Here, too, you can choose between a download as *rpm* package or *tar.gz* archive. The installation is largely identical to the one previously mentioned. Apart from the option of a complete download, IBM also offers you a download split into four (JDK) or three (JRE) files, which will be of interest if your Internet connection is prone to crashing. The individual files then have to be combined prior to installation. This is done with the cat command in the following form:

```
cat [file1] [file2] [file3] [file4] > [outputfile]
```

The IBM Java package stands out in particular for its high speed, since large parts of it are written in C++.

## Kaffe

The "Kaffe" project is attempting to imitate the Java Virtual Machine including the class libraries as Open Source project. This project was created by Tim Wilkinson and is now supported by a great many other Java programmers. Unfortunately, the version numbers of the Kaffe implementation do not correspond to the usual versions from Sun, which is making the categorisation of its project status difficult. At present (Version 1.0.6) Kaffe is in between Java versions 1.1 and 1.2 from Sun. But some functions are still not yet implemented.

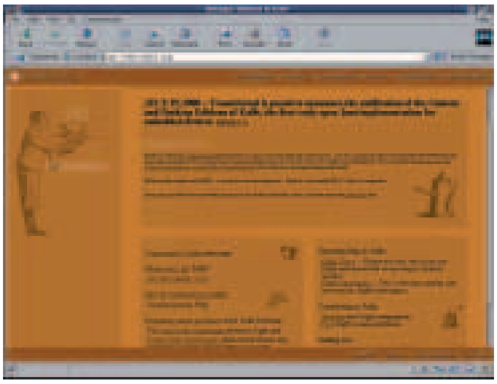


**Figure 3: The Kaffe homepage (http://www.kaffe.org)**

Sadly, installation is not so simple as with the other packages mentioned. The latest release does not work with the current versions of *glibc*, so compilation onto many Linux distributions is not possible. This means you need the latest version from the **CVS** directory. To do this, enter the following commands:

```
cvs -d :pserver:readonly@cvs.kaffe.org:/↩
cvs/kaffe login
cvs -d :pserver:readonly@cvs.kaffe.org:/↩
cvs/kaffe co kaffe
```

After that the source texts of the latest version of Kaffe will be found in the new sub-directory *kaffe*. Compilation is now done with the commands:

```
./configure ––prefix=/usr/lib
make
make install
```

The parameter "––prefix" in the *configure* script specifies the directory in which Kaffe is to be installed. After that Kaffe should be ready to start work.

## Conclusion

As you have seen, there are numerous options for Java programmers and users under Linux. Which alternative best suits your requirements, is something you should try out by testing the various packages. If you want to play safe, it is advisable to turn first to the Sun JDK/JRE. In the next article we will be putting the now-installed JRE into practice and trying out the first Java applications.

**tar.gz** The latest tar versions use the option "-j", to unpack a bzip2-compressed tar archive. For older tar variants this is still "-I" (with a capital I), while for very old ones there is no appropriate option. If your tar reacts to both variants with an error message, decompress the archive with bunzip2.

**CVS** The "Concurrent Versions System" allows all those involved in large programming projects to have write/read access to the source files. At the same time CVS offers the feature of version control, so that current or older versions can be extracted at any time from the CVS "tree".

## URLs

| | |
|---|---|
| JDK and JRE | *http://java.sun.com/j2se* |
| BlackDown Java | *http://www.blackdown.org* |
| IBM | *http://www-106.ibm.com/developerworks/java/jdk* |
| Kaffe | *http://www.kaffe.org* |