NAME

jolt − Java Virtual Machine Daemon

SYNOPSIS

jolt [**-hv**] [**-t** *timeout*] [**-j** *switch* ...] *tool* [*arg* ...]

DESCRIPTION

jolt is a small wrapper program that allows multiple consecutive invocations of the Sun JDK tools **javac**, **javadoc**, and **jar** to run within the same Java Virtual Machine instance, thereby significantly reducing their startup times. **jolt** can also be used to run arbitrary Java programs in the same manner as the **java** launcher program: it loads the specified class and executes its main() method.

The JVM reuse is accomplished by spawning an ‘‘ephemeral’’ daemon which launches the JVM and remains in the background until it times out due to inactivity. While the daemon is running, an invocation of **jolt** results in a call to the daemon (via an IPC mechanism), which runs the specified JDK tool in the JVM. By default, the daemon shuts down the JVM and exits after 60 seconds of inactivity, but this timeout value is configurable.

Invocations of **javac**, **javadoc**, and **jar** through **jolt** behave exactly as if they were called directly: the standard output and standard error streams of **jolt** will produce the corresponding output from the tool, and its exit status will be the exit status returned by the tool. Therefore, **jolt** can be effortlessly integrated with build tools such as GNU **make**, and will not break programs that analyze the output and return status of the JDK tools.

Invocations of Java programs through **jolt** behave exactly as if they were launched with **java**: the standard input, standard output, and standard error streams of the JVM will be connected to those of **jolt**.

For each invocation, the current working directory of the JVM and its *user.dir* system property are adjusted to match that of the shell from which **jolt** was invoked.

jolt is an iterative utility which means that it can only process one Java tool or program invocation at a time. However, multiple users on a system can use **jolt** concurrently; each user will have his or her own instance of the ephemeral daemon and JVM.

A call to **System.gc()** is made after each request is completed, in an effort to minimize memory consumption by the JVM.

OPTIONS

-h Display a command synopsis and exit.

-v Display the program’s version information and exit.

-t *timeout*
Specify a timeout interval for the ephemeral daemon. The default timeout is 60 seconds. If no requests are received within the timeout interval, the ephemeral daemon will shut down the JVM and exit; the next invocation of **jolt** will respawn the daemon, which will restart the JVM. If the value is followed immediately by an ’s’ or by no character at all, the value is interpreted as seconds; if an ’m’ as minutes; and if an ’h’ as hours. The minimum allowed timeout is 10 seconds. The JVM is not shut down if there is more than 1 active thread after the timeout expires.

-j *switch*
Pass the *switch* to the Java virtual machine. If several switches need to be passed, each must be preceded with a **-j** switch. Note that these switches are only significant when the ephemeral daemon is spawned; they have no effect if it is already running. It is also possible to pass switches through the environment; see the following section for details.

ENVIRONMENT

JAVA_HOME An absolute path to a JDK installation (version 1.4.0 or later).

JAVA_ARGS A list of whitespace-separated switches to pass to the Java virtual machine. Note that these switches are only significant when the ephemeral daemon is spawned; they have no effect if it is already running.

CLASSPATH The classpath for the JVM. All directories in the classpath must be specified as absolute paths. This environment variable is only relevant when executing arbitrary Java programs; it is not used for tool invocations.

LD_LIBRARY_PATH
The dynamic linker path, which must contain the paths to the Java runtime shared libraries.

FILES

/tmp/.jolt_* Jolt communication sockets. The suffix consists of a username.

EXAMPLES

The following command line invokes **javac** on all Java source files in the current directory, with a JVM-specific switch:

```
jolt -j -mx50m javac -deprecation *.java
```

The following command line begins execution of a Java program whose entry point class is ‘‘Hello’’, passing the argument ‘‘foo’’ to its main() method. The JVM timeout is set to 10 minutes.

```
jolt -t 10m java Hello foo
```

NOTES

If **jolt** is interrupted by the user with Control-C while a JVM request is in progress, it forcibly kills the ephemeral daemon and then exits. The JVM must be destroyed in this case because there is no robust programmatic means of stopping an executing Java program within the context of the JVM.

Terminating **jolt** with SIGKILL while a JVM request is in progress will result in a situation where both the running Java program and the shell are connected to the terminal’s I/O streams, possibly leading to bizarre behavior. This situation cannot be avoided because SIGKILL cannot be caught, and the ephemeral daemon is not always a child process of the client.

JDK 1.4.0 or later is *required* for **jolt** because some of the programmatic interfaces that it uses were introduced in that version.

Java programs invoked through **jolt** must not call System.exit(); doing so will lead to undefined behavior.

Java is a trademark of Sun Microsystems, Inc.

AUTHOR

Mark Lindner (markl@gnu.org)