Running the RVM

Jikes™ RVM executes Java virtual machine byte code instructions from `.class` files. It does *not* compile Java™ source code. Therefore, you must compile all Java source files into bytecode using your favorite Java compiler.

For example, to run class `foo` with source code in file `foo.java`:

```
% javac foo.java
% rvm foo
```

The general syntax is

```
rvm [rvm options...] class [args...]
```

You may choose from a myriad of options for the `rvm` command-line. Options fall into two categories: *standard* and *non-standard*. Non-standard options are preceded by "**`-X:`**" and differ between virtual machines (e.g. Jikes RVM's options may not be available in HotSpot and vice-versa).

Standard Command-Line Options

We currently support a subset of the JDK 1.5 standard options. Below is a list of all options and their descriptions. Unless otherwise noted each option is supported in Jikes RVM.

Option	Description
{-cp or -classpath} <directories and zip/jar files separated by ":">	set search path for application classes and resources
-D<name>=<value>	set a system property
-verbose:[class | gc | jni]	enable verbose output
-version	print current VM version and terminate the run
-showversion	print current VM version and continue running
-fullversion	like "-version", but with more information
-? or -help	print help message
-X	print help on non-standard options
-jar	execute a jar file
-javaagent:<jarpath>[=<options>]	load Java programming language agent, see java.lang.instrument

Non-Standard Command-Line Options

It is generally the case that the non-standard options may change from one release to another. However, the core and memory non-standard options that are listed here don't change often.

The bulk of the non-standard options are grouped according to the subsystem that they control. See below for details.

Core Non-Standard Command-Line Options

Option	Description
-X:verbose	Print out additional lowlevel information for GC and hardware trap handling
-X:verboseBoot=<number>	Print out additional information while VM is booting, using verbosity level <number>
-X:sysLogfile=<filename>	Write standard error message to <filename>
-X:ic=<filename>	Read boot image code from <filename>

-X:id=<filename>	Read boot image data from <filename>
-X:ir=<filename>	Read boot image ref map from <filename>
-X:vmClasses=<path>	Load the org.jikesrvm.* and java.* classes from <path>
-X:processors=<number|"all">	The number of processors that the garbage collector will use

Memory Non-Standard Command-Line Options

Option	Description
-Xms<number><unit>	Initial size of heap where <number> is an integer, an extended-precision floating point or a hexadecimal value and <unit> is one of T (Terabytes), G (Gigabytes), M (Megabytes), pages (of size 4096), K (Kilobytes) or <no unit> for bytes
-Xmx<number><unit>	Maximum size of heap. See above for definition of <number> and <unit>

Subsystem Non-Standard Command-Line Options

The other non-standard command line options are not listed here because they will change from time to time. To get a current list of these options, you can do the following:

1. To find out which subsystems are available in the image you're using:

```
rvm -X
```

2. To get information about the options for a specific subsystem (e.g. the garbage collection subsystem)

```
rvm -X:gc:help
```

3. To find out the current values of the options for a specific subystem (e.g. the optimizing compiler subsystem)

```
rvm -X:opt:printOptions
```

Running Jikes RVM with valgrind

Jikes RVM can run under valgrind, as of SVN revision 6791 (29-Aug-2007). Applying a patch of this revision to release 3.2.1 should also produce a working system. Versions of valgrind CVS prior to release 3.0 are also known to have worked.

To run a Jikes RVM build with valgrind, use the `-wrap` flag to invoke valgrind, eg

```
rvm -wrap "path/to/valgrind --smc-check=all <valgrind-options>"
<jikesrvm-options> ...
```

this will insert the invocation of valgrind at the appropriate place for it to operate on Jikes RVM proper rather than a wrapper script.

Under some circumstances, valgrind will load shared object libraries or allocate memory in areas of the heap that conflict with Jikes RVM. Using the flag -X:gc:eagerMmapSpaces=true will prevent and/or detect this. If this flag reveals errors while mapping the spaces, you will need to rearrange the heap to avoid the addresses that valgrind is occupying.