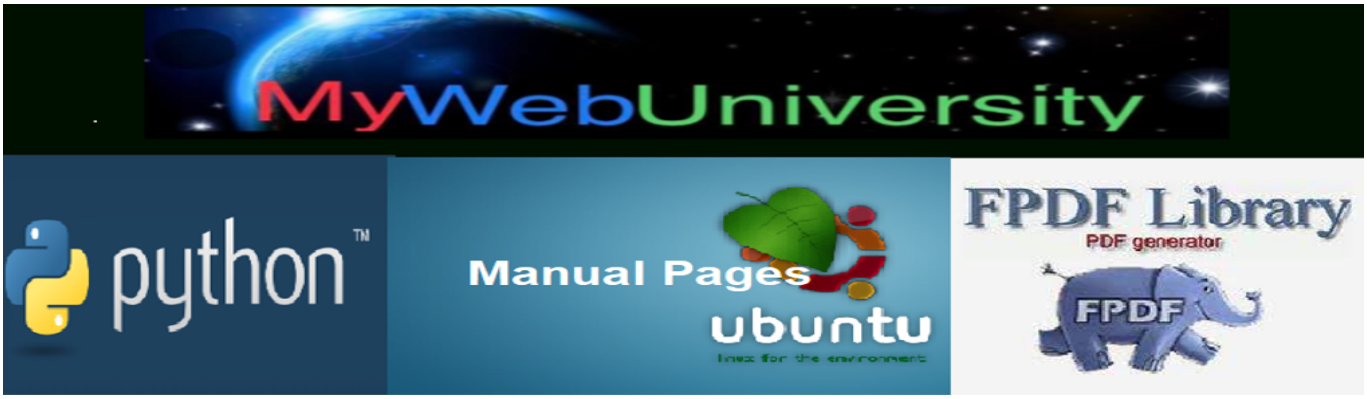

Rocky Enterprise Linux 9.2 Manual Pages on command 'jlink.1'

$ man jlink.1

JLINK(1) JDK Commands JLINK(1)

NAME

jlink - assemble and optimize a set of modules and their dependencies into a custom run? time image

SYNOPSIS

jlink [options] --module-path modulepath --add-modules module [, module...]

options

Command-line options separated by spaces. See jlink Options.

modulepath

The path where the jlink tool discovers observable modules. These modules can be modular JAR files, JMOD files, or exploded modules.

module The names of the modules to add to the runtime image. The jlink tool adds these modules and their transitive dependencies.

DESCRIPTION

The jlink tool links a set of modules, along with their transitive dependences, to create a custom runtime image.

Note:

Developers are responsible for updating their custom runtime images.

JLINK OPTIONS

--add-modules mod [, mod...]

Adds the named modules, mod, to the default set of root modules. The default set of root modules is empty.

--bind-services

Link service provider modules and their dependencies.

-c ={0|1|2} or --compress={0|1|2}

Enable compression of resources:

? 0: No compression

? 1: Constant string sharing

? 2: ZIP

--disable-plugin pluginname

Disables the specified plug-in. See jlink Plug-ins for the list of supported plug-ins.

--endian {little|big}

Specifies the byte order of the generated image. The default value is the format of your system's architecture.

-h or --help

Prints the help message.

--ignore-signing-information

Suppresses a fatal error when signed modular JARs are linked in the runtime image. The signature-related files of the signed modular JARs aren't copied to the runtime image.

--launcher command=module or --launcher command=module/main

Specifies the launcher command name for the module or the command name for the module and main class (the module and the main class names are separated by a slash (/)).

--limit-modules mod [, mod...]

Limits the universe of observable modules to those in the transitive closure of the named modules, mod, plus the main module, if any, plus any further modules specified in the --add-modules option.

--list-plugins

Lists available plug-ins, which you can access through command-line options; see jlink Plug-ins.

-p or --module-path modulepath

Specifies the module path.

If this option is not specified, then the default module path is $JAVA_HOME/jmods. This directory contains the java.base module and the other standard and JDK mod?

ules. If this option is specified but the java.base module cannot be resolved from it, then the jlink command appends $JAVA_HOME/jmods to the module path.

--no-header-files

Excludes header files.

--no-man-pages

Excludes man pages.

--output path

Specifies the location of the generated runtime image.

--save-opts filename

Saves jlink options in the specified file.

--suggest-providers [name, ...]

Suggest providers that implement the given service types from the module path.

--version

Prints version information.

@filename

Reads options from the specified file.

An options file is a text file that contains the options and values that you would typically enter in a command prompt. Options may appear on one line or on several lines. You may not specify environment variables for path names. You may comment out lines by prefixing a hash symbol (#) to the beginning of the line.

The following is an example of an options file for the jlink command:

```
#Wed Dec 07 00:40:19 EST 2016
--module-path mlib
--add-modules com.greetings
--output greetingsapp
```

JLINK PLUG-INS

Note:

Plug-ins not listed in this section aren't supported and are subject to change.

For plug-in options that require a pattern-list, the value is a comma-separated list of elements, with each element using one the following forms:

? glob-pattern

? glob:glob-pattern

? regex:regex-pattern

? @filename

? filename is the name of a file that contains patterns to be used, one pattern per line.

For a complete list of all available plug-ins, run the command jlink --list-plugins.

Plugin compress

Options

--compress={0|1|2}[:filter=pattern-list]

Description

Compresses all resources in the output image.

? Level 0: No compression

? Level 1: Constant string sharing

? Level 2: ZIP

An optional pattern-list filter can be specified to list the pattern of files to include.

Plugin include-locales

Options

--include-locales=langtag[,langtag]*

Description

Includes the list of locales where langtag is a BCP 47 language tag. This option supports locale matching as defined in RFC 4647. Ensure that you add the module jdk.localedata when using this option.

Example:

--add-modules jdk.localedata --include-locales=en,ja,*-IN

Plugin order-resources

Options

--order-resources=pattern-list

Description

Orders the specified paths in priority order. If @filename is specified, then each line in pattern-list must be an exact match for the paths to be ordered.

Example:

--order-resources=/module-info.class,@classlist,/java.base/java/lang/

Plugin strip-debug

Options

--strip-debug

Description

Strips debug information from the output image.

Plugin generate-cds-archive

Options

--generate-cds-archive

Description

Generate CDS archive if the runtime image supports the CDS feature.

JLINK EXAMPLES

The following command creates a runtime image in the directory greetingsapp. This command links the module com.greetings, whose module definition is contained in the directory mlib.

```
jlink --module-path mlib --add-modules com.greetings --output greetingsapp
```

The following command lists the modules in the runtime image greetingsapp:

```
greetingsapp/bin/java --list-modules
com.greetings
java.base@11
java.logging@11
org.astro@1.0
```

The following command creates a runtime image in the directory compressedrt that's stripped of debug symbols, uses compression to reduce space, and includes French language locale information:

```
jlink --add-modules jdk.localedata --strip-debug --compress=2 --include-locales=fr --output compressedrt
```

The following example compares the size of the runtime image compressedrt with fr_rt, which isn't stripped of debug symbols and doesn't use compression:

```
jlink --add-modules jdk.localedata --include-locales=fr --output fr_rt
du -sh ./compressedrt ./fr_rt
23M     ./compressedrt
36M     ./fr_rt
```

The following example lists the providers that implement java.security.Provider:

```
jlink --suggest-providers java.security.Provider
Suggested providers:
  java.naming provides java.security.Provider used by java.base
```

java.security.jgss provides java.security.Provider used by java.base

java.security.sasl provides java.security.Provider used by java.base

java.smartcardio provides java.security.Provider used by java.base

java.xml.crypto provides java.security.Provider used by java.base

jdk.crypto.cryptoki provides java.security.Provider used by java.base

jdk.crypto.ec provides java.security.Provider used by java.base

jdk.crypto.mscapi provides java.security.Provider used by java.base

jdk.security.jgss provides java.security.Provider used by java.base

The following example creates a custom runtime image named mybuild that includes only ja?va.naming and jdk.crypto.cryptoki and their dependencies but no other providers. Note that these dependencies must exist in the module path:

jlink --add-modules java.naming,jdk.crypto.cryptoki --output mybuild

The following command is similar to the one that creates a runtime image named greet?ingsapp, except that it will link the modules resolved from root modules with service binding; see the Configuration.resolveAndBind method.

jlink --module-path mlib --add-modules com.greetings --output greetingsapp --bind-services

The following command lists the modules in the runtime image greetingsapp created by this command:

greetingsapp/bin/java --list-modules

com.greetings

java.base@11

java.compiler@11

java.datatransfer@11

java.desktop@11

java.logging@11

java.management@11

java.management.rmi@11

java.naming@11

java.prefs@11

java.rmi@11

java.security.jgss@11

java.security.sasl@11

java.smartcardio@11

java.xml@11

java.xml.crypto@11

jdk.accessibility@11

jdk.charsets@11

jdk.compiler@11

jdk.crypto.cryptoki@11

jdk.crypto.ec@11

jdk.crypto.mscapi@11

jdk.internal.opt@11

jdk.jartool@11

jdk.javadoc@11

jdk.jdeps@11

jdk.jfr@11

jdk.jlink@11

jdk.localedata@11

jdk.management@11

jdk.management.jfr@11

jdk.naming.dns@11

jdk.naming.rmi@11

jdk.security.auth@11

jdk.security.jgss@11

jdk.zipfs@11

org.astro@1.0