

MEMORYSCAPE

**COMMAND-LINE
OPTIONS**



VERSION 2.4.0



Copyright © 2007–2008 by TotalView Technologies, LLC.
Copyright © 1999–2008 by Etnus LLC. All rights reserved.
Copyright © 1998–1999 by Etnus, Inc.
Copyright © 1996–1998 by Dolphin Interconnect Solutions, Inc.
Copyright © 1993–1996 by BBN Systems and Technologies, a division of BBN Corporation.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of TotalView Technologies LLC. (TotalView Technologies).

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

TotalView Technologies has prepared this manual for the exclusive use of its customers, personnel, and licensees. The information in this manual is subject to change without notice, and should not be construed as a commitment by TotalView Technologies. TotalView Technologies assumes no responsibility for any errors that appear in this document.

TotalView and TotalView Technologies are registered trademarks of TotalView Technologies LLC.

TotalView uses a modified version of the Microline widget library. Under the terms of its license, you are entitled to use these modifications. The source code is available at <http://www.totalviewtech.com/Products/TotalView/developers>.

All other brand names are the trademarks of their respective holders.

Contents

1 MemoryScape Command-Line Options

Invoking MemoryScape.....	1
Syntax	1
Options	2

MemoryScape Command-Line Options

This chapter presents the commands you use to invoke MemoryScape as well as the variables you can place in a `.memrc` file.

Invoking MemoryScape

This section describes the structure (syntax) of the `memscape` and `memscript` commands that you use to invoke MemoryScape.

- Use the `memscape` command to invoke the MemoryScape GUI.
- Use the `memscript` command to invoke MemoryScape in batch mode.

Topics in this section are:

- Syntax
- "Options" on page 2

Syntax

```
{ memscript | memscape } [ filename [ corefile ] ] [ options ]
```

Arguments

<i>filename</i>	Specifies the pathname of the executable being debugged. This can be an absolute or relative pathname. The executable must be compiled with debugging symbols turned on, normally the <code>-g</code> compiler option. Any multiprocess programs that call <code>fork()</code> , <code>vfork()</code> , or <code>execve()</code> should be linked with the <code>dbfork</code> library.
<i>corefile</i>	Specifies the name of a core file. Use this argument in addition to <i>filename</i> when you want to examine a core file with MemoryScape.

The MemoryScape debugger is a source-level debugger with a motif-based graphic user interface and features for debugging distributed programs, multiprocess programs, and multithreaded programs. MemoryScape is available on a number of different platforms.

If you specify mutually exclusive options on the same command line (for example, `-ccq` and `-nccq`), MemoryScape uses the last option that you enter.

Options

- a** *args* Passes all subsequent arguments (specified by *args*) to the program specified by *filename*. This option must be the last one on the command line.
- aix_use_fast_trap** Tells MemoryScape that it support the AIX fast trap mechanism. You must either set this option on the command line or place it within a `.memrc` file.
- bg** *color* Same as **-background**.
- compiler_vars** Some Fortran compilers (HP f90/f77, HP f90, SGI 7.2 compilers) output debugging information that describes variables the compiler itself has invented for purposes such as passing the length of character*(*) variables. By default, MemoryScape suppresses the display of these compiler-generated variables.
However, you can specify the **-compiler_vars** option to display these variables. This is useful when you are looking for a corruption of a runtime descriptor or are writing a compiler.
- no_compiler_vars**
(Default) Tells MemoryScape that it should not show variables created by the Fortran compiler.
- control_c_quick_shutdown**
-ccq (Default) Tells MemoryScape to kill attached processes and exit.
- no_control_c_quick_shutdown**
-nccq Invokes code that sometimes allows MemoryScape to better manage the way it kills parallel jobs when it works with management systems. This has only been tested with SLURM. It may not work with other systems.
- debug_file** *consoleoutputfile*
Redirects MemoryScape console output to a file named *consoleoutputfile*.
- Default:** All MemoryScape console output is written to `stderr`.

- display** *displayname*
Sets the name of the X Windows display to *displayname*. For example, **-display vinnie:0.0** will display MemoryScape on the machine named "vinnie."
- Default:** The value of your `DISPLAY` environment variable.
- dump_core** Allows MemoryScape to dump a core file of itself when an internal error occurs. This is used to help TotalView Technologies debug MemoryScape problems.
- no_dumpcore**
(Default) Does not allow MemoryScape to dump a core file when it gets an internal error.
- env** *variable=value*
Tells MemoryScape to add an environment variable to the environment variables passed to your program by the shell. If the variable already exists, it effectively replaces the previous value. You need to use this command for each variable being added; that is, you cannot add more than one variable with an `env` command.
- nptl_threads** Tells MemoryScape that your application is using NPTL threads. You only need to use this option if MemoryScape cannot determine that which thread package your program is using.
- no_nptl_threads**
Tells MemoryScape that you are not using the NPTL threads package. Use this option if MemoryScape thinks your application is using it and it isn't.
- pid** *pid*
Tells MemoryScape to attach to process *pid* after it starts executing.
- search_path** *pathlist*
Specifies a colon-separated list of directories that MemoryScape will search when it looks for source files. For example:
`memscape --search_path proj/bin:proj/util`

-signal_handling_mode "action_list"

Modifies the way in which MemoryScape handles signals. You must enclose the *action_list* string in quotation marks to protect it from the shell.

An *action_list* consists of a list of *signal_action* descriptions separated by spaces:

```
signal_action[ signal_action] ...
```

A signal action description consists of an action, an equal sign (=), and a list of signals:

```
action=signal_list
```

An *action* can be one of the following: **Error**, **Stop**, **Resend**, or **Discard**.

A *signal_specifier* can be a signal name (such as **SIGSEGV**), a signal number (such as 11), or a star (*), which specifies all signals. We recommend that you use the signal name rather than the number because number assignments vary across UNIX sessions.

The following rules apply when you are specifying an *action_list*:

- (1) If you specify an action for a signal in an *action_list*, MemoryScape changes the default action for that signal.
- (2) If you do not specify a signal in the *action_list*, MemoryScape does not change its default action for the signal.
- (3) If you specify a signal that does not exist for the platform, MemoryScape ignores it.
- (4) If you specify an action for a signal more than once, MemoryScape uses the last action specified.

If you need to revert the settings for signal handling to MemoryScape's built-in defaults, use the **Defaults** button in the **File > Signals** dialog box.

For example, here's how to set the default action for the **SIGTERM** signal to resend:

```
"Resend=SIGTERM"
```

Here's how to set the action for **SIGSEGV** and **SIGBUS** to error, the action for **SIGHUP** to resend, and all remaining signals to stop:

```
"Stop=* Error=SIGSEGV,SIGBUS \  
Resend=SIGHUP"
```

- shm "action_list"** Same as **-signal_handling_mode**.
- stderr pathname** Names the file to which your program will write **stderr** information while executing within MemoryScape. If the file exists, MemoryScape overwrites it.
- stderr_append** Tells MemoryScape to append the information it writes to **stderr** to the file named in the **-stderr** command or the file named in GUI or a MemoryScape variable. If the file does not exist, MemoryScape creates it.
- stderr_is_stdout pathname** Tells MemoryScape to redirect **stderr** to **stdout**.
- stdin pathname** Names the file from which your program will read information while executing within MemoryScape.
- stdout pathname** Names the file to which your program will write **stdout** information while executing within MemoryScape. If the file exists, MemoryScape overwrites it.
- stdout_append** Tells MemoryScape to append the information it writes to **stdout** to the file named in the **-stdout** option or the file named in a MemoryScape variable. If the file does not exist, MemoryScape creates it.
- verbosity level** Sets the verbosity level of MemoryScape-generated messages to *level*, which can be one of **silent**, **error**, **warning**, or **info**.

Default: info

Index

A

-a option to memscape command 2
aix_use_fast_trap command-line option 2
-aix_use_fast_trap option 2

B

bg command-line option 2
-bg option 2

C

ccq command-line option 2
command-line options
 -aix_use_fast_trap 2
 -bg 2
 -ccq 2
 -control_c_quick_shutdown 2
 -debug_file 2
 -display 2
 -dump_core 2
 -env 2
 -nccq 2
 -no_compiler_vars 2
 -no_control_c_quick_shutdown 2
 -no_dumpcore 2

 -no_nptl_threads 2
 -nptl_threads 2
 -pid 2
 -search_path 3
 -shm 3
 -signal_handling_mode 3
 -stderr 3
 -stderr_append 3
 -stderr_is_stdout 3
 -stdin 3
 -stdout 3
 -stdout_append 3
 -verbosity 3
console output redirection 2
control_c_quick_shutdown command-line option 2
core
 dumping for MemoryScape 2

D

debug_file command-line option 2
-debug_file option 2
display command-line option 2
-display option 2
dump_core command-line option 2

E

-env command-line option 2

H

handling signals 3

M

MemoryScape
 command options 2

N

-nccq option 2
-no_compiler_vars option 2
-no_control_c_quick_shutdown option 2
-no_dump_core option 2
no_dumpcore command-line option 2
no_nptl_threads command-line option 2
nptl_threads command-line option 2

O

options
 -aix_use_fast_trap 2

P

pid command-line option 2

V

S

search_path command-line option 3
-shm option 3
signal_handling_mode command-line option 3
-signal_handling_mode option 3
signals, handling in MemoryScape 3
SLURM, control_c_quick_shutdown variable 2
-stderr command-line option 3
-stderr_append command-line option 3
stderr_append command-line option 3
stderr_is_stdout command-line option 3
stdin command-line option 3
-stdout command-line option 3
stdout_append command-line option 3

V

-verbosity option 3