RogueWave SOFTWARE
Innovate with Confidence

MemoryScape Cheat Sheet

1 Compiling Programs. Compile your programs using the –g option. For example:
gcc -g –o *my_prog my_prog.c*

2 Starting MemoryScape. One way is to type

memscape *my_prog* **–a** *arguments*

on the command line.

Or, from the shell, type **memscape** to open the MemoryScape window.

- Select **Add new program**.
- Use the **Next** buttons to move through the screens to set up and start your memory debugging session.

3 Checking for Errors. MemoryScape stops program execution and raises an event flag before events such as the following occur:

- Freeing memory that is already freed
- Freeing the wrong address
- Freeing an interior pointer
- Misaligning blocks

Click on the event flag for details.

4 Backtraces Defined. When your program makes a memory request, MemoryScape records the stack frames that existed when the action occurred. This list of frames is called a *backtrace*.

5 Showing Memory Leaks

- Press the **Halt** button to stop program execution.
- Select the Memory Reports tab, then Leak Detection.
- Within the Leak Detection page, select either Source or Backtrace Report.

If there are leaks, MemoryScape summarizes the number of leaks and how much memory is associated with a backtrace or source line.

6 Displaying the Heap Graphically. Use the Heap Status Graphical Report to see how your program is using memory. Clicking on a block in the top area displays information at the bottom. Clicking on the Backtrace/Source tab and selecting a backtrace highlights the related blocks.

7 Filtering Information. To reduce the amount of data displayed, you can filter information related to a process, library, source file, class name, line number, etc.

- Select **Tools > Filters...** Menu Item, or **Manage Filters** on the left.
- Select the **Add** button to create a filter. Creating a filter is similar to creating a message filter within an email program.
- After creating the filter, generate a report by clicking on the button. 

8 Tracking Memory Usage. You can track how much memory your program is using by generating Memory Usage Reports.

9 Block Painting. Block painting helps you locate problems caused by accessing allocated memory before you initialize it, or accessing deallocated memory.

Block painting writes a bit pattern into newly allocated or deallocated blocks. When you see this pattern, you know that a problem is occurring.

Enable block painting by selecting **Paint Memory** within the Memory Debugging Options Page.

10 Tracking Deallocations

- On the Heap Status Graphical Report, right-click on a selected block and select **Properties**.
- Select **Hide Backtrace Information**, then expand the block by clicking **+**.
- At the bottom of the expanded window, select **Notify when deallocated**.

11 Tracking Memory Blocks. The Block Properties Window can contain information about many memory blocks. After you place a block in the window, it's often hard to identify the allocation. Adding a comment lets you remember why you're tracking a block.

12 Comparing Memory Use. You can use the **Export Memory Data** link on the left to write memory information to disk. At a later time, use the **Add Memory Debugging File** link on the left of the Home page to bring the information back into MemoryScape. You can examine this information in exactly the same way as normal memory information, or by using the Memory Compare page.

13 Guarding Allocated Memory. Guards detect when a program writes beyond the limits of your memory block. To turn them on, either select **Medium** from Basic Memory Debugging Options or select **Guard allocated memory** from Advanced Memory Debugging Options.

With guards on, MemoryScape adds a small segment of memory before and after each block that you allocate. Here are two ways to find corrupted memory blocks:

- When the program frees the memory, the guards are checked for corruption. If a corrupted guard is found, MemoryScape stops program execution and raises an event flag.
- Select **Corrupted Memory Report** from the Memory Reports page.

14 Using Red Zones. Red Zones allow MemoryScape to immediately notify you if your program oversteps the bounds of your allocated block. Turn them on by selecting **High** from Basic Memory Debugging Options, or by selecting **Use Red Zones to find memory access violations** from Advanced Memory Debugging Options.

With Red Zones on, a page of memory is placed either before or after your allocated block, and if your program tries to read or write in this zone, MemoryScape stops program execution and raises an event flag. Click on the event flag to see the event details.

15 If You Have Trouble Running Your Program in MemoryScape.

- For AIX, read Chapter 4 of the user guide "Debugging Memory Problems Using MemoryScape" at **www.roguewave.com/support/product-documentation.aspx**.
- If you're running a program that spawns processes, the problem may be that your environment isn't sending environment variables to the process. If this happens, you'll need to explicitly add libraries that we provide. Again, please see Chapter 4 of the MemoryScape user guide.

Using MemoryScape from TotalView® for HPC

1 Starting MemoryScape from TotalView

- Select the **Debug > Enable Memory Debugging** command before you tell your program to start executing.

  If you don't do this, memory debugging won't work.
- Let your program run, then stop it after it allocates some memory.
- Select **Debug > Open MemoryScape**.
- Select a report, such as Leak Detection or Heap Graphical Report.

If you make changes or run your program to another breakpoint, you'll need to regenerate the report.

2 Identifying Dangling Pointers. When memory debugging is enabled and TotalView displays the value of a variable, it tells you if memory is allocated or if you're looking at a dangling pointer.

3 Seeing Changes (Setting a Baseline)

- In a Process Window, select the **Debug > Heap Baseline > Set Heap Baseline** command.
- Run your program. After stopping execution, select **Debug > Heap Baseline > Heap Change Summary** to see any memory allocations or leaks that occurred since you set the baseline.

4 Comparing Memory Use. If you created a baseline, go to the MemoryScape window and select your memory report. You can select the option Relative to Baseline, which shows you the information relative to the baseline you set.