Kmemleak

Contents

Stable: 16.10.2020 - 12:24 / Revision: 16.10.2020 - 12:24

The content format pdf is not supported by the content model wikitext.

Return to Main Page.

Stable: 17.11.2021 - 16:46 / Revision: 17.11.2021 - 15:58

You do not have permission to edit this page, for the following reasons:

- The action you have requested is limited to users in one of the groups: Administrators, Editors, Reviewers, Selected_editors, ST_editors.
- The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

---

You can view and copy the source of this page.

== Article purpose == This article provides the basic information needed to start using the Linux kernel tool: '''kmemleak'''<ref name=procodedotorg/>. == Introduction == {{: Trace_and_debug_tools_assignment_table_template}} <onlyinclude> | [[Kmemleak|kmemleak]] | [[:Category: Monitoring tools|Monitoring tools]] | style="text-align:left;" | '''kmemleak'''<ref name=procodedotorg>http://www. procode.org/kmemleak/</ref> provides a means to detect possible kernel memory leaks in a similar way to a tracing garbage collector, with the difference that the orphan objects are not freed, but only reported via /sys/kernel /debug/kmemleak. | {{N}} || {{Y}} || {{Y}} | {{N}} || {{N}} || {{Y}} |- </onlyinclude> |} == Installing the trace and debug tool on your target board == In order to use '''kmemleak''', the Linux kernel configuration must activate CONFIG_DEBUG_KMEMLEAK: Symbol: '''DEBUG_KMEMLEAK''' Location: Kernel Hacking ---> Memory Debugging --> {{highlight|[*] Kernel memory leak detector}} '''For {{EcosystemRelease | revision=2.0.0 | range=and after}}''': In this ecosystem release, this is possible to enable/disable automatic kmemleak scan at boot up.<br> This configuration is activated by default when DEBUG_KMEMLEAK configuration is set. Symbol: '''DEBUG_KMEMLEAK_AUTO_SCAN''' Location: Kernel Hacking ---> Memory Debugging --> [*] Kernel memory leak detector {{highlight|Enable kmemleak auto scan thread on boot up}} <div class="mw-collapsible mw-collapsed"> '''For {{EcosystemRelease | revision=1.1.0 | range=and before}}''': <div class="mw-collapsible-content" > Since memory may be allocated or freed before kmemleak is initialised, an early log buffer is used to store these actions.<br> In this ecosystem release, the default buffer size is limited, and this is recommended to increase it. {{Warning|If kmemleak reports ''"early log buffer exceeded"'' and debugfs entry is not present, you can increase the log buffer size by changing the configuration value, for example to 5000 Symbol: '''DEBUG_KMEMLEAK_EARLY_LOG_SIZE''' [&#61;400] Location: {{Green|Range: [200 40000]}} Kernel Hacking ---> Memory Debugging --> [*] Kernel memory leak detector {{highlight|(400) Maximum kmemleak early log entries}} }} </div></div> === Using STM32MPU Embedded Software Distribution === ==== Developer Package ==== To enable '''CONFIG_DEBUG_KMEMLEAK''' in the Linux kernel configuration, please refer to the [[Menuconfig or how to configure kernel]] article to find instructions for modification of the configuration and recompiling Linux kernel image in Developer Package context. ==== Distribution Package ==== To enable '''CONFIG_DEBUG_KMEMLEAK''' in the Linux kernel configuration, please refer to the [[Menuconfig or how to configure kernel]] article to find instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context. === Using STM32MPU Embedded Software Distribution for Android&trade; === ==== Distribution Package ==== To enable '''CONFIG_DEBUG_KMEMLEAK''' in the Linux kernel configuration, please refer to the [[How to customize kernel for Android]] article to find instructions for modification of the configuration and recompiling Linux kernel image in Distribution Package context. == Getting started == {{Android | Need to enable root access rights *Using ADB shell is ADB link available: {{PC$}} adb root {{PC$}} adb shell {{Board$}} ... *Using uart console shell: {{Board$}} su {{Board$}} ... }} === Reading kmemleak report === A kernel thread scans the memory every 10 minutes (by default) and prints the number of new unreferenced objects found. To display the details of all the possible memory leaks: {{Board$}} cat /sys/kernel/debug/kmemleak ''Note'': debugfs is mounted by default, otherwise you can mount it using the following command. Please refer to the [[Debugfs]] article. Kmemleak result example: contains an extract of the memory content, and backtrace of function calls, to help you when debugging. unreferenced object 0xed638800 (size 64): comm "swapper/0", pid 1, jiffies 4294937542 (age 197.490s) hex dump (first 32 bytes): 01 00 00 00 77 f9 ff 7a cf 37 dd e3 01 00 00 00 ....w.. z.7...... 00 92 63 ed 40 00 00 00 00 00 00 00 01 00 00 00 ..c.@........... backtrace: [<c048d130>] pinconf_generic_parse_dt_config+0x10c/0x13c [<c0490f88>] stm32_pctrl_dt_node_to_map+0x90/0x3f4 [<c048ca58>] pinctrl_dt_to_map+0x130/0x35c [<c04895a0>] create_pinctrl+0x60/0x3b0 [<c0489a08>] devm_pinctrl_get+0x38/0x68 [<c0582a24>] pinctrl_bind_pins+0x48/0x280 [<c055f7f8>] driver_probe_device+0xc0

/0x470 [<c055fca8>] __driver_attach+0x100/0x11c [<c055db08>] bus_for_each_dev+0x4c/0x9c [<c055ecc4>] bus_add_driver+0x1c0/0x264 [<c0560910>] driver_register+0x78/0xf4 [<c0101c48>] do_one_initcall+0x44/0x168 [<c0f00e74>] kernel_init_freeable+0x1c0/0x24c [<c0a65ac4>] kernel_init+0x8/0x110 [<c0108b50>] ret_from_fork+0x14/0x24 [<ffffffff>] 0xffffffff === Trigerring an intermediate memory scan === {{Board$}} echo scan > /sys/kernel/debug/kmemleak === Clearing the list of all current possible memory leaks === {{Board$}} echo clear > /sys/kernel/debug/kmemleak === All kmemleak commands === Memory scanning parameters can be modified at run-time by writing to the <code>/sys/kernel/debug/kmemleak</code> file. The following parameters are supported: '''off''' - disable kmemleak (irreversible) '''stack=on''' - enable the task stacks scanning (default) '''stack=off''' - disable the tasks stacks scanning '''scan=on''' - start the automatic memory scanning thread (default) '''scan=off''' - stop the automatic memory scanning thread '''scan=<secs>''' - set the automatic memory scanning period in seconds (default 600, 0 to stop the automatic scanning) '''scan''' - trigger a memory scan '''clear''' - clear list of current memory leak suspects, done by marking all current reported unreferenced objects in grey, or freeing all kmemleak objects if kmemleak is disabled. '''dump=<addr>''' - dump information about the object found at <addr> == To go further == If enabled in the Linux kernel configuration, Kmemleak can also be object found at <addr> == To go further == If enabled in the Linux kernel configuration, Kmemleak can also be disabled at boot time by passing {{highlight|kmemleak{{=}}off}} on the kernel command line. Conversely, if '''CONFIG_DEBUG_KMEMLEAK_DEFAULT_OFF''' is enabled in the Linux kernel configuration, kmemleak is disabled by default. Symbol: '''DEBUG_KMEMLEAK_DEFAULT_OFF''' Location: Kernel Hacking ---> Memory Debugging --> [*] Kernel memory leak detector {{highlight|[*] Default kmemleak to off}} Passing {{highlight|kmemleak{{=}}on}} on the kernel command line enables the function. == References == <references /> * Useful external links {| ! scope=col | Document link ! scope=col | Document Type ! scope=col | Description |- | [https://www.kernel.org/doc/html/latest/dev-tools/kmemleak.html Kernel Memory Leak Detector] | Standard | Documentation from kernel.org |} {{ArticleBasedOnModel | Trace and debug tools article model }} {{PublicationRequestId | 9880 | 07Dec'18 }} [[Category:Linux monitoring tools]]

Templates used on this page:

- Template:Highlight (view source)
- Template:Info (view source)
- Template:STDarkBlue (view source)

Return to Main Page.