Linux tracing, monitoring and debugging

Stable: 15.10.2019 - 10:06 / Revision: 15.10.2019 - 10:04

Contents

1 Article purpose

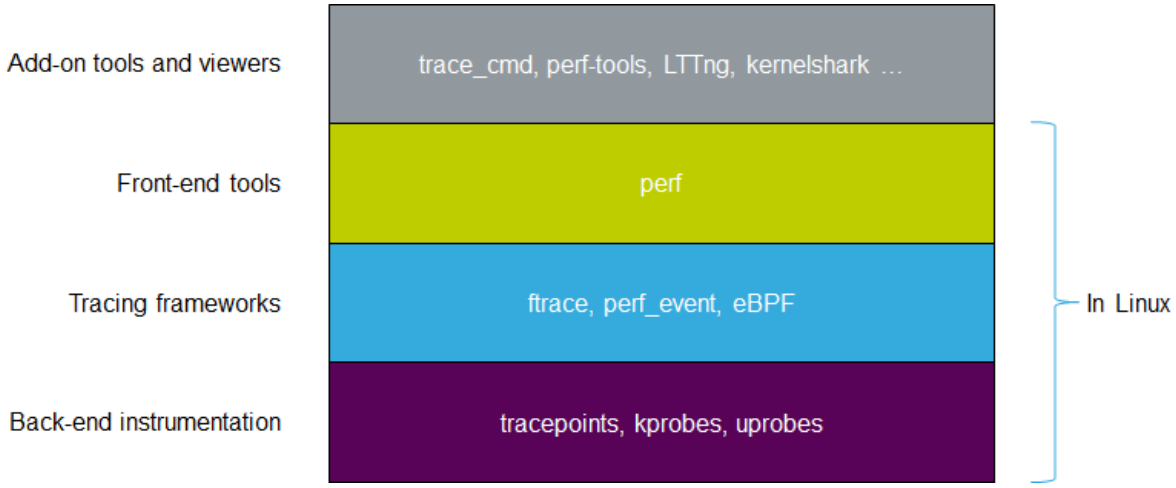
This article provides useful information to start using Linux® tracing, monitoring and debugging environments.

Two entry points are proposed in this article:

- Linux tracing, monitoring and debugging tools, which gives an overview of some Linux® tools including usage and application domain. **This chapter is useful when you already know the domain or the interface to search for**.
- Trace and debug overview per Linux software frameworks, which points to articles explaining how to get trace and debug information about the Linux® software frameworks that are relevant for the STM32MPU Embedded Software. **This chapter is useful when you know the Linux® framework to search for**.

2 Linux trace architecture overview

The Linux® trace architecture can be organized into four levels as shown in the figure below (*inspired by Brendan Gregg presentation*[1]):



2.1 Back-end instrumentation

The back-end instrumentation provides tracing sources built in the Linux® kernel. They are split into three categories:

- **tracepoints**: kernel static tracing, statically placed at logical places in the kernel. It provides key event details as a "format" string.
- **kprobes**: kernel dynamic tracing. It allows to trace function calls, returns and line numbers.
- **uprobes**: dynamic user-level tracing.

2.2 Tracing framework

Also named tracers, they use tracing sources.

Tracing frameworks include kernel in-tree tracers such as ftrace and perf_events, and out-of-tree tracers such as SystemTap and sysdig.

2.3 Front-end tools

Front-end tools come on top of tracers and help to configure them. For example:

- trace-cmd or LTTng for ftrace
- perf or perf-Tools for perf_events

2.4 Add-on tools and viewer

Add-on tools are also on top of tracers. However, they are not embedded inside the Linux® kernel.

Viewer tools propose Visual interpretation of trace data. For example:

- **kernelshark** for ftrace/trace-cmd
- **Trace Compass**[2] for LTTng (and more)
- **Flame Graph**[3] for perf

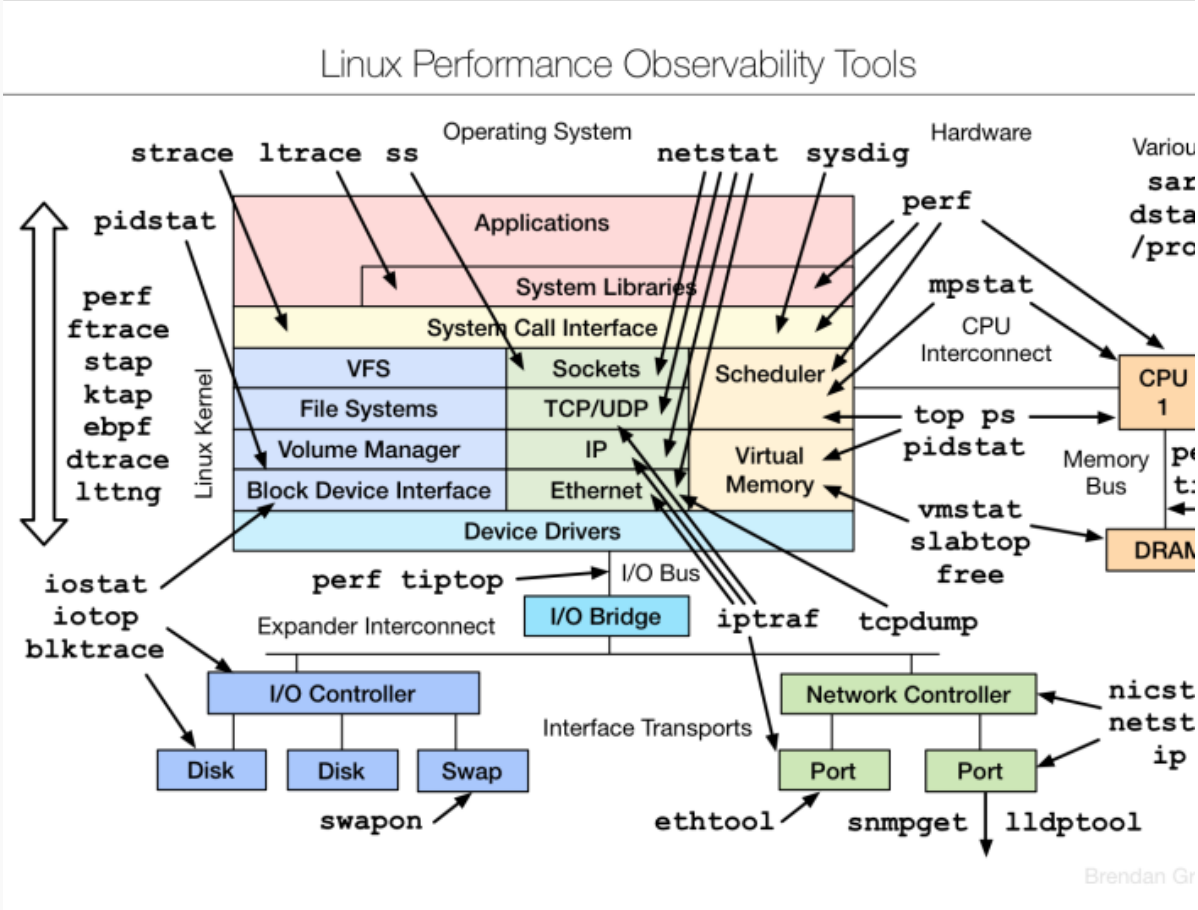
3 Linux tracing, monitoring and debugging tools

Linux® provides many tools that are either dedicated to one function or multifunction (generic).

They cover both Linux® kernel and Linux® user space.

3.1 Domain mapping

The following mapping, done by Brendan Gregg [4], shows the different existing tools associated to the different Linux® frameworks.



Note: The above image has been created by **Brendan Gregg** (Netflix) and can be found on his official web site.

3.2 Tool overview

The following table provides a brief description of the tool, as well as its availability depending on the software packages:

✅: this tool is either present (ready to use or to be activated), or can be integrated and activated on the software package.

: this tool is not present and cannot be integrated, or it is present but cannot be activated on the software package.

Tool			STM32MPU Embedded Software distribution			STM32MPU Embedded Software distribution for Android™		
Name	**Category**	**Purpose**	**Starter Package**	**Developer Package**	**Distribution Package**	**Starter Package**	**Developer Package**	**Distribution Package**
blktrace	Tracing tools	**blktrace**[5] generates traces of the I/O traffic on block devices (SD card, USB, eMMC...)	✗	✓	✓	✗	✗	✓
systemd core dump	Debugging tools	**systemd core dump**: generates core dump files on Linux	✓	✗	✓	✗	✗	✗
ethtool	Monitoring tools	**ethtool**[6] allows to query or control network driver and hardware	✓	✓	✓	✗	✗	✗

		settings, in particular for wired Ethernet devices.						
ftrace	Tracing tools	**ftrace**[7] (Function Tracer) is a powerful kernel tracing utility that is able, for instance, to trace every kernel function calls and kernel events without adding any extra code in your kernel source code	✗	✗	✓	✗	✗	✓
		The GNU Project debugg	✗*	✓	✗**	✗	✗	✓**

GDB	Debugging tools	er, **GDB** [8], allows monitoring program execution, or what the program was doing at the moment it crashed.	** Cross compile gdb and openocd binaries are required and only available from Developer Package.* *** It is recommended to use the Developer Package to run the gdb debug session, which provided all dependencies*			** Cross compile gdb and openocd binaries are required and only available from Distribution Package.*		
ifconfig	Monitoring tools	**ifconfig**[9] is a system administration utility for network interface configuration. **ifconfig is deprecated and has been replaced by *ip*** (A web page provide	✓	✓	✓	✓	✓	✓

		s a comparison between ifconfig and ip [10])						
ip	Monitoring tools	**ip**[11] shows / manipulates routing, devices, policy routing and tunnels of network interfaces. **ip replaces the deprecated command ifconfig**	✓	✓	✓	✓	✓	✓
		kmemleak[12] provides a means to detect possible kernel memory leaks in a similar way to						

kmemleak	Monitoring tools	a tracing garbage collector, with the difference that the orphan objects are not freed, but only reported via /sys/kernel/debug/kmemleak.	✗	✓	✓	✗	✗	✓
		trace-cmd[13] command interacts with the Ftrace tracer that is built inside the Linux kernel. It interfaces with the Ftrace specific files found						

trace-cmd	Tracing tools	in the debugfs file system under the tracing directory. **kernelshark**[14] is a front-end reader of trace-cmd output. "trace-cmd record" and "trace-cmd extract" create a trace.dat (trace-cmd.dat) file. kernelshark can read this file, and produce a graph	✗	✗	✓	✗	✗	✗

		and list view of the corresponding data.						
ltrace	Tracing tools	**ltrace**[15] is used to display the calls to shared libraries made by a userspace application. **ltrace** is a userspace application. *Its use is very similar to strace.*	✗	✗	✓	✗	✗	✗
		LTTng[16] is an open source tracing framework for Linux kernel and user spaces. It is a						

LTTng	Tracing tools	powerful tool that can be used for many purposes. LTTng traces need to be processed /displayed with a host tool such as Trace Compass[17], based on Eclipse plugin for increased portability.	✗	✗	✓	✗	✗	✗
		netdata[18] is a system for distributed real-time performance and health monitori						

netdata	Monitoring tools	ng. It provides unparalleled insights, in real-time, of everything happening on the system it runs (including applications such as web and database servers), using modern interactive web dashboards.	✓	✓	✓	✗	✗	✗
netstat	Monitoring tools	**netstat** [19] prints network connections, routing tables, interface statistics, masquerade	✓	✓	✓	✓	✓	✓

		connections, and multicast membership information.						
perf	Monitoring tools	**perf**[20] is a Linux user space tool, which allows getting system performance figures	✓	✓	✓	✗*	✗*	✗*
						*Note: **simpleperf**[21] is present as equivalent but with less options*		
strace	Tracing tools	**strace**[22] is able to intercept and record the system calls which are called by a process and the signals which are received by another process.	✓	✓	✓	✓	✓	✓

Coming soon

sysprof Monitoring tools **sysprof**[23] is a statistical, system-wide profiler for Linux. It helps in finding the functions in which a program spends most of its time.

sysprof proposes a user interface available directly on the board display screen.

sysstat Monitoring tools The **sysstat** [24] tool suite contains utilities to monitor the system performance and usage activity.

It contains various utilities, common to many commercial Unix distributions, as well as tools that can be scheduled (via a scheduler such as cron) to collect and historize performance and activity data:

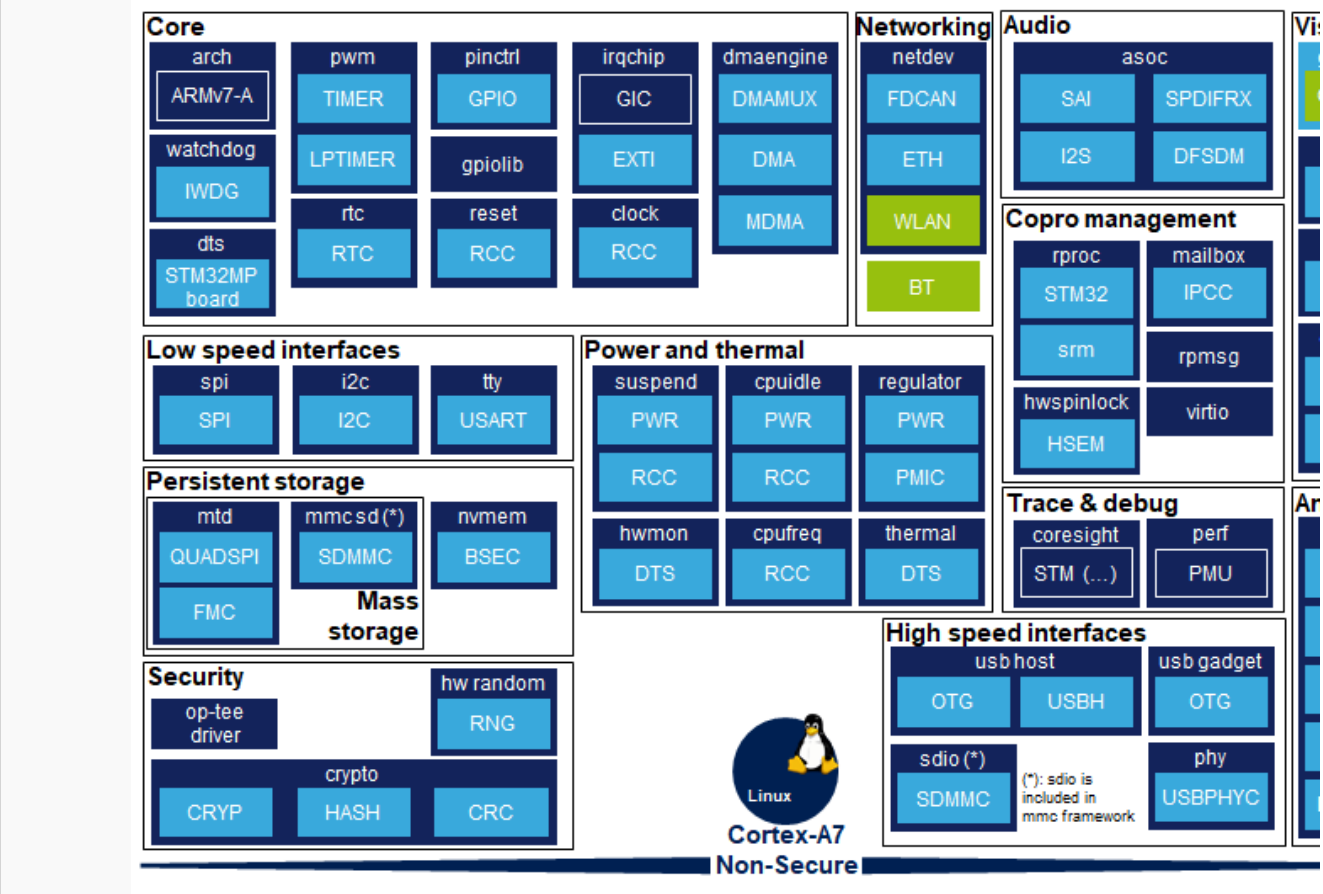
- **iostat**: reports CPU statistics and input/output statistics for block devices and partitions.
- **mpstat**: reports individual or combined processor related statistics.
- **pidstat**: reports statistics for Linux tasks (processes): I/O, CPU, memory, etc.
- **sar**: collects, reports and saves system activity information (CPU, memory, disks, interrupts, network interfaces, TTY, kernel tables,etc.)
- **sadf**: displays data collected by sar in multiple formats (CSV, XML, JSON, etc.). This command can also be used to exchange data with other programs or to draw graphs illustrating the various activities collected by sar using SVG (Scalable Vector Graphics) format.

tcpdump Monitoring tools **tcpdump**[25] is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is connected. top Monitoring tools The **top**[26] program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of tasks currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for tasks are all user configurable and that configuration can be made persistent across restarts. (*Extracted from man page*[26]) valgrind Monitoring tools **valgrind**[27] is an instrumentation framework for building dynamic analysis tools. Some Valgrind tools can automatically detect many memory management and threading bugs, and profile your programs in detail.

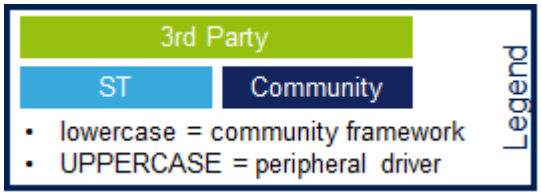
This is tool for Linux application analysis.

4 Trace and debug overview per Linux software frameworks

The picture below allows accessing to different Linux software frameworks which provide specific trace and debug information in their "***How to trace and debug the framework***" dedicated chapter.



STM32MP15 Linux kernel overview



5 Tips

How to find Linux kernel driver associated to a device.

How to use the kernel dynamic debug.

6 Documentation and web articles

A lot of articles on the web mention Linux® kernel tracing and profiling. The following links provide a good introduction to these topics:

- Linux Performance Analysis - New Tools and Old Secrets: description of the Linux® technology and of the different tools available.

- Yocto project: Tracing and profiling: How to enable tracing and profiling tools using Yocto
- Brendan Gregg Linux performance page

More general Linux performance information are available in nice slideshare presentation (Brendan Gregg), on the Brendan Gregg official web site or in LinuxCon2014 article.

Reference list:

1. ↑ http://www.brendangregg.com/linuxperf.html
2. ↑ http://www.tracecompass.org
3. ↑ http://www.brendangregg.com/flamegraphs.html
4. ↑ http://www.brendangregg.com/linuxperf.html
5. ↑ https://linux.die.net/man/8/blktrace
6. ↑ https://linux.die.net/man/8/ethtool
7. ↑ https://elinux.org/Ftrace
8. ↑ https://www.gnu.org/software/gdb
9. ↑ https://linux.die.net/man/8/ifconfig
10. ↑ https://tty1.net/blog/2010/ifconfig-ip-comparison_en.html
11. ↑ https://linux.die.net/man/8/ip
12. ↑ http://www.procode.org/kmemleak/
13. ↑ https://lwn.net/Articles/410200/
14. ↑ http://rostedt.homelinux.com/kernelshark
15. ↑ https://www.ltrace.org/
16. ↑ http://lttng.org
17. ↑ http://tracecompass.org/
18. ↑ https://my-netdata.io
19. ↑ https://linux.die.net/man/8/netstat
20. ↑ https://perf.wiki.kernel.org/index.php/Main_Page
21. ↑ https://source.android.com/devices/tech/debug/eval_perf
22. ↑ https://strace.io/
23. ↑ http://www.sysprof.com/
24. ↑ http://sebastien.godard.pagesperso-orange.fr/
25. ↑ http://www.tcpdump.org/
26. ↑ [26.0] [26.1] http://linux.die.net/man/1/top
27. ↑ http://valgrind.org/