Linux Trace Toolkit Viewer User Guide

Mathieu Desnoyers

This document describes how to install Linux Trace Toolkit Viewer and how to use it.

Table of Contents

Next
Introduction

Chapter 1. Introduction

Linux Trace Toolkit (LTT) is a tracing tool that permits to get all the possible execution information from the Linux Kernel. It is based on kernel instrumentation and a high-speed relay file system to copy the information from the kernel space to the user space.

Linux Trace Toolkit Viewer (LTTV) is the second generation of visualization tool. It is based on a trace format (the files where the data is recorded on disk) written by the LTTng tracer.

This document explains all the steps that are necessary in order to record a trace with LTT and view it with LTTV.

Chapter 2. Getting started

2.1. Installing LTTng and LTTV

Follow the QUICKSTART guide found at ltt.polymtl.ca.

At this point, LTTV is installed in the default directory. You may find the lttv executable in /usr/local/bin and the librairies in /usr/local/lib. You will also notice the presence of the convert executable in /usr/local/bin. This tool will be used later to convert from the Linux Trace Toolkit trace format to the LTTV format.

2.2. Running the executable with basic libraries

Starting the graphical mode with the basic viewer activated is as simple as :

```
$ lttv-gui
```

Using the text mode is very simple too. Look in /usr/local/lib/lttv/plugins for the list of modules. You may use the --help switch to get basic help on the command line parameters of every loaded modules. To simply output the events of a trace in a text file, try the textDump module. The batchAnalysis module permits to do batch mode analysis (state and statistics calculation) on a trace.

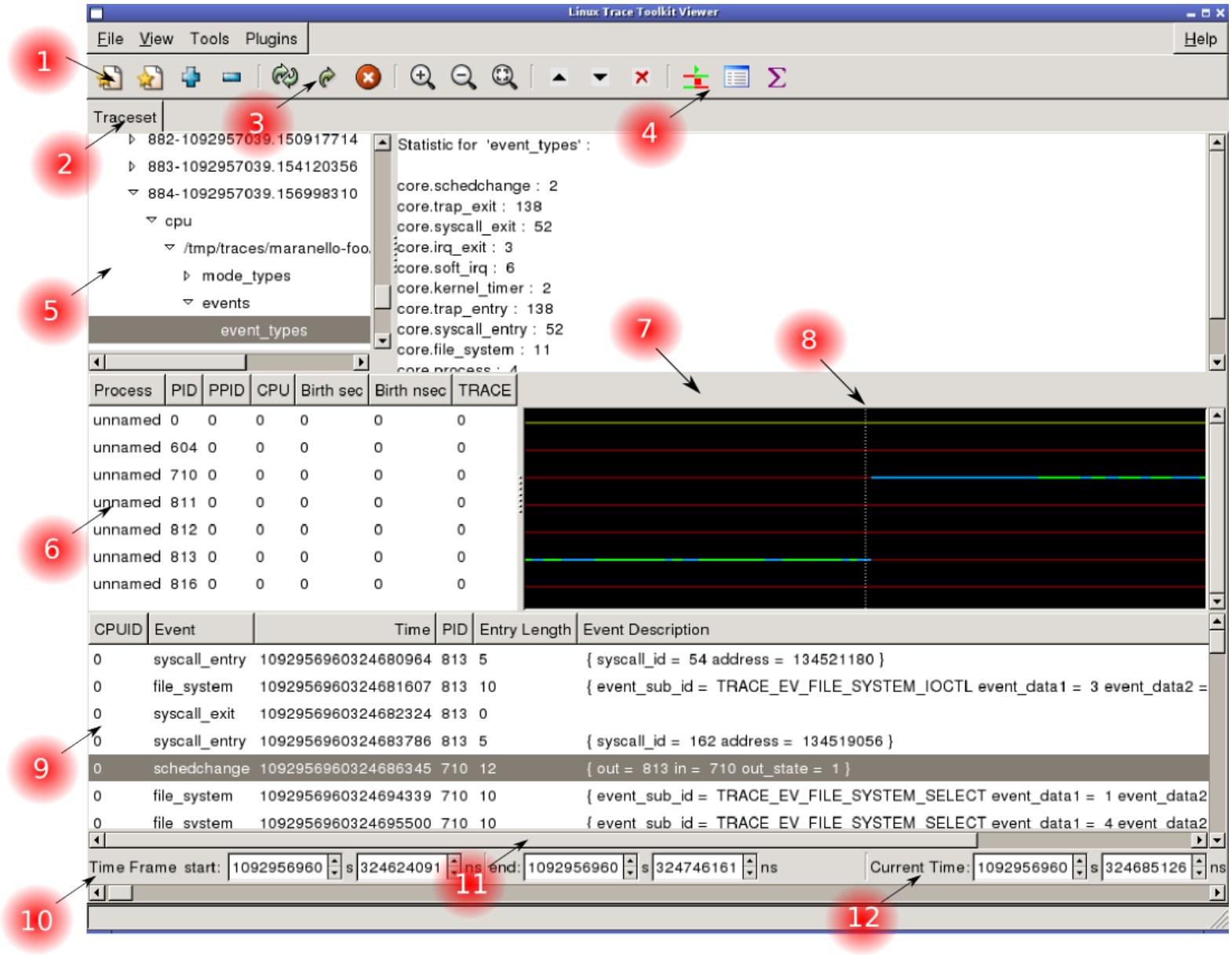
```
$ lttv -L /usr/local/lib/lttv/plugins -m textDump --help
```

Chapter 3. Using LTTV graphical interface

3.1. LTTV main window

This section describes the main functionnalities that are provided by the LTTV GUI and how to use them.

By default, when the lttv GUI starts with all the graphical modules loaded, it loads the statistics viewer, the control flow viewer, and the detailed event list inside a tab. Other viewers can be added later to this tab by interacting with the main window. Let's describe the operations available on the window :



Linux Trace Toolkit Viewer GUI

1. This toolbar allows you to navigate through the basic functionnalities of LTTV. The first button opens a new window and the second one, a new tab. You can leave your mouse over the buttons to read the information provided by the tooltips.

2. This notebook, containing different tabs, lets you select the "Trace Set" you want to interact with. A trace set is an aggregation of traces, synchronised in time. You may also want to use one tab per viewer by simply cloning the traceset to a new tab. This way, you can have vertically stacked viewers in one tab, as well as different viewers, independant from the time interval. Note that once the Trace Set cloning is done, each trace set becomes completely independant. For Traceset cloning, see the File Menu.

3. These buttons let you control the computation in progress on a trace. As sometimes the computation may last for a while, you may want to stop it, restart it from the beginning or simply to continue from where you stopped. This is exactly what those three buttons offer you.

4. Buttons on the right side of the last spacer are semantically different from the others. While the other buttons at the left side of the bar are built in the lttv program and let you operate the basic functionnalities, the buttons at the right side let you add a viewer to the active Tab. They belong to the viewers themselves. The number of buttons that appears there should directly depend on the number of viewer's modules loaded.

5. This is a tree representing the multiple statistics available for the current traceset. This is shown by the guistatistics viewer.

6. This is the Y axis of the guicontrolflow viewer. It shows the process list of the traced system. You may notice that it grows : it dynamically adds process when they appear in the trace.

7. This is a (missing) time bar for the X axis. Maybe will it be used for viewer specific buttons eventually. Work in progress.

8. The is the current time selected. The concept of current event and current time selected is synchronised in a Tab for all the viewers. The control flow viewer shows it a vertical white dotted line. You move this marker by clicking on the background of the process state graph. This graph shows evolution of each process's state through time. The meaning of the colors will be explained later.

9. This is the details event list. It shown the detailed information about each event of the trace. It is synchronised with the current time and current event, so selecting an event changes other viewer's current time and reciprocally.

10. You can enter the values of start time and end time you wish to see on the screen here. It also supports pasting time as text input, simply by clicking of the "Time Frame", "start" or "end:" fields. A valid entry consists of any digital input separated by any quantity of non digital characters. For example : "I start at 356247.124626 and stop at 724524.453455" would be a valid input for the "Time Frame" field.

11. This horizontal scrollbar modifies the window of time shown by all the viewers in the tab. It is linked with the fields below it (described at number 10 and 12). Another way to modify the time shown is to use the zoom buttons of the toolbar (yes, the ones that looks like magnifying glasses).

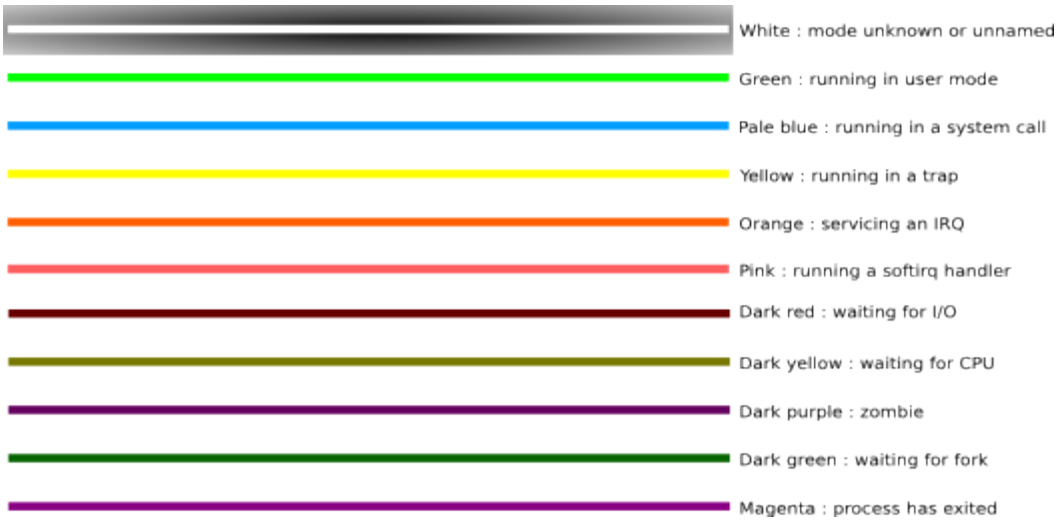
12. This field works just like the "Time Frame" field. It modifies the current time selected by the viewers. For example, changing its value will change the current event selected by the detailed events list and the current time selected by the control flow viewer.

Prev
Running the executable with basic libraries

Home

Next
Control Flow View Colors

3.2. Control Flow View Colors



Control Flow View Color Legend

Here is a description of the colors used in the control flow view. Each color represents a state of the process at a given time.

- White : this color is used for process from which state is not known. It may happen when you seek quickly at a far time in the trace just after it has been launched. At that moment, the precomputed state information is incomplete. The "unknown" state is used to identify this. Note that the viewer gets refreshed once the precomputation ends.
- Green : This color is only used for process when they are running in user mode. That includes execution of all the source code of an executable as well as the libraries it uses.
- Pale blue : A process is doing a system call to the kernel, and the mode is switched from process limited rights to super user mode. Only code from the kernel (including modules) should be run in that state.
- Yellow : The kernel is running a trap that services a fault. The most frequent trap is the memory page fault trap : it is called every time a page is missing from physical memory.
- Orange : IRQ servicing routine is running. It interrupts the currently running process. As the IRQ does not change the currently running process (on some architectures it uses the same stack as the process), the IRQ state is shown in the state of the process. IRQ can be nested : a higher priority interrupt can interrupt a lower priority interrupt.
- Pink : SoftIRQ handler is running. A SoftIRQ is normally triggered by an interrupt that whishes to have some work done very soon, but not "now". This is especially useful, for example, to have the longest part of the network stack traversal done : a too long computation in the interrupt handler would increase the latency of the system. Therefore, doing the long part of the computation in a softirq that will be run just after the IRQ handler exits will permits to do this work while interrupts are enabled, without increasing the system latency.
- Dark red : A process in that state is waiting for an input/output operation to complete before it can continue its execution.

- Dark yellow : A process is ready to run, but waiting to get the CPU (a schedule in event).
- Dark purple : A process in zombie state. This state happens when a process exits and then waits for the parent to wait for it (wait() or waitpid()).
- Dark green : A process has just been created by its parent and is waiting for first scheduling.
- Magenta : The process has exited, but still has the control of the CPU. It may happend if it has some tasks to do in the exit system call.

Prev Using LTTV graphical interface | Home Up | Next Using LTTV text modules

Chapter 4. Using LTTV text modules

4.1. The batch analysis module

This batch analysis module can be invoked like this :

```
$ lttv -L path/to/lib/plugins -m batchAnalysis\
-t trace1 -t trace2 ...
```

It permits to call any registered action to perform in batch mode on all the trace set, which consists of the traces loaded on the command line. Actions that are built in the batchAnalysis module are statistics computation. They can be triggered by using the -s (--stats) switch.

However, the batchAnalysis module is mostly a backend for every other text module that does batch computation over a complete trace set.

4.2. The text dump module

The goal of this module is to convert the binary data of the traces into a formatted text file.

The text dump module is a good example of a usage of the batch analysis module backend. In fact, the text dump module depends on it. You don't need to explicitly load the batchAnalysis module though, as lttv offers a rich module backend that deals with the dependencies, loading the module automatically if needed.

The text dump module is invoked just like the batchAnalysis module. It adds more options that can be specified in argument. You may specify the -o switch for the output file name of the text dump. You can enable the output of the field names (the identifier of the fields) with the -l switch. The -s switch, for process states, is very useful to indicate the state in which the process is when the event happens.

If you use the --help option on the textDump module, you will see all the detail about the switches that can be used to show per cpu statistics and per process statistics. You will notice that you can use both the switches for the batchAnalysis module and those for textDump. You will also notice that the options --process_state (from textDump) and --stats (from batchAnalysis) has the same short name "-s". If you choose to invoke this option using the short name, it will use the option of the last module loaded just before the -s switch.

For exemple, if you load the textDump module with -m textDump, it will first load the batchAnalysis module, and then load itself. As it is the last module loaded, the -s switch used after it will signify --process_stats. On the other hand, if you choose to specify explicitly the loading of both modules like this :

```
$ lttv -L path/to/lib/plugins -m batchAnalysis -s\
-m textDump -s -t trace
```

The first "-s" will invoke batchAnalysis --stats and the second "-s" will invoke textDump --process_state. The list of options generated by --help follows the order of registration of the options by the modules, therefore the invocation order of the modules.

Linux Trace Toolkit Viewer User Guide

Mathieu Desnoyers

This document describes how to install Linux Trace Toolkit Viewer and how to use it.

Table of Contents

Next
Introduction