ftrace for Android hackers.

What ftrace is, why you should be using it, and how to get the most out of it.

Mihir Mehta

Kernel Team
Systems Core Group
SRI Noida
mihir.mehta@samsung.com

6 January 2014

Overview

Overview of the material.

1. Introduction to ftrace
2. Filesystems and debugfs
3. ftrace functionality
4. Example

Introduction to ftrace

What is ftrace anyway?

- Most commonly known as the function tracer.
- Idea: save timing information for the execution of kernel functions and export the information to userspace.
- Utility: This is useful to kernel developers who need to keep an eye on how much time particular functions are taking.
- Today: Expanded to include tracers for many other kinds of info.
- Output: variety of formats, including human readable output - immensely useful for embedded developers (that means us!)

Filesystems and debugfs - introduction

- Linux - several different filesystems, intended for different purposes.
- Common features - encapsulated in the VFS (Virtual Filesystem Switch).
- This allows most filesystems to be mounted using the mount command.
- Our focus: debugfs.
- Like sysfs and procfs - intended to allow kernel developers to export information to userspace and get control instructions from userspace.
- Different from sysfs and procfs - both of these have specific purposes and their indiscriminate use for debugging is discouraged. Not so for debugfs.

Filesystems and debugfs - using debugfs

- When you need debugfs in your build, you need to configure it into your kernel, using the configuration option `CONFIG_DEBUG_FS`.
  - NOTE: this option is automatically selected when you select any option that enables ftrace during kernel configuration.
  - NOTE: Regardless of ftrace, debugfs is usually enabled by default, because many kernel subsystems have come to depend upon it.
- When debugfs is configured, the directory `/sys/kernel/debug` is created. debugfs is usually mounted into this directory.
- This mounting can be done, either by adding a line to the `/etc/fstab` file, or by using the mount command manually.
- Once this mounting is done, we can interact with ftrace, which stores its files in the `tracing` directory at the root of debugfs.

ftrace functionality

- ftrace offers several different tracers:
  - `function`
  - `function_graph`
  - `irqsoff`
  - `preemptoff`
  - `preemptirqsoff`
  - `wakeup`
  - `wakeup_rt`
  - `nop`
- Any one of these can be active at a time.
- To change the active tracer, write its name into the file `tracing/current_tracer`.
- The tracer `nop` simply disables all tracers when it is written to `tracing/current_tracer`.
- We shall restrict our further discussion to `function` and `function_graph`.

ftrace functionality

- The tracer `function` traces all kernel functions. Probes functions on their entry. Configured in by `CONFIG_FUNCTION_TRACER`.
- The tracer `function_graph` traces all kernel functions. Probes functions on their entry as well as on their exit. Optionally, provides a graph of function calls. Configured in by `CONFIG_FUNCTION_GRAPH_TRACER`.
- Both these tracers rely on run-time patching, which is a technique whereby certain parts of the kernel executable are overwritten.
- More specifically, addresses for function calls are overwritten, so that some profiling data can be collected before the actual function is called.
- This is done only for the selected functions.