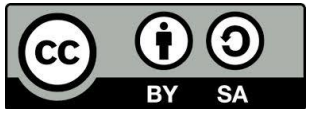
Introducción a Ftrace

Arquitectura Interna de Linux - 2016



J.C. Sáez
High Performance Computing Division
ArTeCs Group
Universidad Complutense de Madrid

Ftrace

- Herramienta de depuración/inspección del kernel Linux
- Se usa realizando lecturas y escrituras en un conjunto de entradas en el sistema de ficheros *debugfs*
  - Directorio: `/sys/kernel/debug/tracing`
  - Sólo el usuario `root` puede configurar/usar `ftrace`
- Consta de un conjunto de *tracers*
  - `nop`
  - `function`
  - `function_graph`
  - ...
- Documentación de ftrace
  - `https://www.kernel.org/doc/Documentation/trace/ftrace.txt`

Montar debugfs para poder usar ftrace

- Las entradas de ftrace sólo estáran accesibles si:
  1. Kernel compilado con soporte de ftrace (`CONFIG_FTRACE=y`)
  2. ... y debugfs está montado
     - `mount -t debugfs nodev /sys/kernel/debug`

Montando debugfs

```
kernel@debian:~$ sudo -i
[sudo] password for kernel:
root@debian:~# ls /sys/kernel/debug/tracing
ls: cannot access /sys/kernel/debug/tracing: No such file or directory
root@debian:~# ls
root@debian:~# mount -t debugfs nodev /sys/kernel/debug
root@debian:~# cd /sys/kernel/debug/tracing
root@debian:/sys/kernel/debug/tracing# ls
README                      current_tracer           instances
printk_formats      set_graph_function  trace_pipe available_events
dyn_ftrace_total_info     kprobe_events    saved_cmdlines
set_graph_notrace   trace_stat
available_filter_functions  enabled_functions        kprobe_profile
set_event           trace                tracing_cpumask
available_tracers           events                   max_graph_depth
...
```

Entradas básicas de Ftrace

- Entradas básicas en `/sys/kernel/debug/tracing`

Entrada	Descripción
`tracing_on`	Permite activar/desactivar ftrace o consultar estado actual. Escribir la cadena "1" para activar ftrace o "0" para desactivarlo.
`trace`	Al leer de esta entrada se muestran los mensajes almacenados en los *buffers* de ftrace (un *buffer* por CPU)
`trace_pipe`	Similar a `trace`, pero además los buffers se vacían al mostrar su contenido (semántica productor/consumidor)
`available_tracers`	Lista el conjunto de *tracers* disponibles
`current_tracer`	Permite consultar/modificar el *tracer* activo leyendo/escribiendo en la entrada
`available_filters`	Lista el conjunto de funciones del kernel o de los módulos cargados que pueden "filtrarse" al usar el tracer `function`.
`set_ftrace_filter`	Permite establecer la función (o funciones) para las que ftrace insertará un mensaje en el buffer cuando éstas sean invocadas.

`nop` tracer y `trace_printk()`

- Tracer por defecto en el kernel
- Captura únicamente los mensajes que el kernel o los módulos imprimen con la función `trace_printk()`

`trace_printk()`

- Para usar `trace_printk()` desde un módulo del kernel ...
  - `#include <linux/ftrace.h>`
- Uso similar a `printf()`, pero mensaje se inserta en buffer interno de ftrace
- Mucho más eficiente que `printk()`. Además, si ftrace está desactivado, no tiene efecto (modo *silencioso*)

Ejemplo de uso de nop tracer (1/4)

- Modificaremos el módulo de ejemplo clipboard para que muestre un mensaje con trace_printk() y capturaremos la salida con ftrace

Adiciones en clipboard.c (en verde)

```
#include <linux/vmalloc.h>
#include <asm-generic/uaccess.h>
#include <linux/ftrace.h>
...
static ssize_t clipboard_write(struct file *filp, const char __user *
    buf, size_t len, loff_t *off) {
...

  clipboard[len] = '\0'; /* Add the `\0' */
  *off+=len;            /* Update the file pointer */

  trace_printk("Current value of clipboard: %s\n",clipboard);

  return len;
}
...
```

Ejemplo de uso de nop tracer (2/4)

- Compilar y cargar el módulo

Terminal

```
kernel@debian:/tmp/FicherosP1/Clipboard$ make
make -C /lib/modules/3.14.1.lin/build M=/tmp/FicherosP1/Clipboard
make[1]: Entering directory `/usr/src/linux-headers-3.14.1.lin'
  CC [M]  /tmp/FicherosP1/Clipboard/clipboard.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/FicherosP1/Clipboard/clipboard.mod.o
  LD [M]  /tmp/FicherosP1/Clipboard/clipboard.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.14.1.lin'
kernel@debian:/tmp/FicherosP1/Clipboard$ sudo insmod clipboard.ko
[sudo] password for kernel:
kernel@debian:/tmp/FicherosP1/Clipboard$
```

Ejemplo de uso de `nop` tracer (3/4)

- Abriremos 2 terminales
  - (Primer terminal - root)
    1. Asegurarse que ftrace y nop tracer activos
    2. Leer de la entrada `trace_pipe` (bloqueante)
  - (Segundo terminal)
    1. Escribir la cadena "Test" en la entrada `/proc/clipboard`
    2. Escribir la cadena "Something" en la entrada `/proc/clipboard`
- Las acciones realizadas en el segundo terminal harán que se muestren mensajes por el primero (salida de `cat trace_pipe`)

ArTeCS

Ejemplo de uso de nop tracer (4/4)

Terminal 1

```
root@debian:/sys/kernel/debug/tracing# cat current_tracer
nop
root@debian:/sys/kernel/debug/tracing# cat tracing_on
1
root@debian:/sys/kernel/debug/tracing# cat trace_pipe
            bash-5098  [000] ....  1065.269409: clipboard_write:
            Current value of clipboard: Test

            bash-5098  [000] ....  1100.023458: clipboard_write:
            Current value of clipboard: Something
```

Terminal 2

```
kernel@debian:/tmp/FicherosP1/Clipboard$ echo "Test" > /proc/clipboard
kernel@debian:/tmp/FicherosP1/Clipboard$ echo "Something" > /proc/clipboa
kernel@debian:/tmp/FicherosP1/Clipboard$
```

ArTeCS

Tracer `function`

- Vuelca un "mensaje" en el buffer de ftrace cuando se ejecuta cierta función del kernel
  - Permite ver qué funciones se invocan sin modificar el código del kernel (o de un módulo)
- Soporta filtros de funciones
  - Escribir nombre(s) de funcion(es) a trazar en `set_ftrace_filter`
  - El listado de funciones que pueden seleccionarse se puede obtener leyendo de la entrada `available_filter_functions`
- Por defecto, no hay ningún filtro → ¡¡Se trazan todas las funciones (mucha sobrecarga)!!
  - Aconsejable desactivar temporalmente ftrace (`tracing_on`) hasta que se establezcan correctamente los filtros de funciones

Ejemplo de uso del tracer `function` (1/2)

- Usaremos ftrace para que nos avise cuándo se invoca la función `clipboard_read()` del módulo `clipboard`
  - No es preciso modificar el código para esto

Pasos (desde `/sys/kernel/debug/tracing`)

1. Desactivar temporalmente ftrace
   $ echo 0 > tracing_on
2. Activar function tracer y comprobar que se activó correctamente:
   $ echo function > current_tracers ; cat current_tracer
   function
3. Preparar filtros de ftrace
   $ echo clipboard_read > set_ftrace_filter
4. Activar ftrace
   $ echo 1 > tracing_on

Ejemplo de uso del tracer `function` (2/2)

- Una vez configurado el tracer `function`, abrir 2 terminales
  - (Primer terminal - root)
    - Leer de la entrada `trace_pipe` (bloqueante)
  - (Segundo terminal)
    - Leer de la entrada `/proc/clipboard`

Terminal 1

```
root@debian:/sys/kernel/debug/tracing# cat trace_pipe
    cat-6932  [000] ....  3166.842845: clipboard_read <-proc_reg_read
    cat-6932  [000] ....  3166.844485: clipboard_read <-proc_reg_read
```

Terminal 2

```
kernel@debian:/tmp/FicherosP1/Clipboard$ cat /proc/clipboard
Something
kernel@debian:/tmp/FicherosP1/Clipboard$
```

¿Por qué `clipboard_read()` se invoca 2 veces?

Licencia

Arquitectura Interna de Linux - Introducción a Ftrace
Versión 0.3

Este documento (o uno muy similar) está disponible en
`https://cv4.ucm.es/moodle/course/view.php?id=70009`