# QEMU--Trace

## 1. Trace event

1. 在源码目录的每个文件夹中都可以在`trace-event`文件中声明一组静态的trace event

```
(base) [root@node0 migration]# ls -la | grep trace-events
-rw-r--r--.   1 1000 1000  19770 Apr 29 00:49 trace-events
```

2. 所有包含`trace-event`文件的子文件夹都必须被列在源码根目录的`Makefile.objs`的`trace-event-subdirs`项中

```
trace-events-subdirs =
trace-events-subdirs += accel/kvm
trace-events-subdirs += backends
trace-events-subdirs += monitor
// ...
trace-events-subdirs += migration
// ...
```

这样在编译时，被列出的子文件夹的`trace-event`文件就会被`tracetool`脚本处理，生成相关代码

3. 在子文件夹中，关于`trace`的文件会被自动生成

```
-rw-r--r--.   1 root root 268880 Aug 13 09:35 trace.h # 跟踪事件的宏定义内联函数等
-rw-r--r--.   1 root root 101737 Aug 13 09:35 trace.c # 跟踪事件的一些状态声明
```

3. 位于子文件中的`.c`源码会包含`trace.h`文件，另外一些共享的`trace`会在顶层文件夹中的`trace-event`文件中定义，但顶层文件生成`trace-root.h`文件

```
// ...
#include "qemu/thread.h"
#include "trace.h"
#include "exec/target_page.h"
// ...
```

```
-rw-r--r--.   1 root root   22440 Aug 13 09:35 trace-root.c
-rw-r--r--.   1 root root   65014 Aug 13 09:35 trace-root.h
```

4. `Trace ecent`分类
    1. `Nop`编译器会把trace events全部优化掉，这样可以做到没有性能的损失
    2. `Log`后端（默认）直接将trace events输出到标准错误`stderr`，这就相当于把`trace events`都转化为了`debug`的`printf`，这是最简单的后端，而且可以和原本的使用`DPRINTF()`的代码一起使用。
    3. `Simpletrace`后端支持一般的使用场景，并且就在QEMU的源码树中。它可能不像特定平台或者第三方追踪后端那样强大，但是它一定是可移植的。

4. `Ftrace`后端将trace数据写到`ftrace marker`中。这相当于将`trace events`发送到`ftrace环状缓冲区`中，然后你可以拿`qemu`的`trace`数据和`kernel`的`trace`数据(尤其是应用`KVM`时的`kvm.ko`内核模块)作比照来看了。
5. `Syslog`后端用`POSIX`的`syslog API`发送`trace events`，日志被以特定的`LOG_DAEMON`设备或`LOG_PID`选项打开(所以`events`会被打上生成它们的`QEMU`进程的`pid`标签)。所有的`events`会被日志记录在`LOG_INFO`级别。
6. `System-stap`

5. 增加`trace-event`

1. 找到文件夹中的`trace-events`文件

```
// ...
migrate_set_state ""
// 下面这条是自己加的
migrate_set_state_user ""
// 要符合一个特定的语法
migaate_send_rp_recv_bitmap(char *name, int64_t size) "block '%s' size
0x%"PRIi64
// ...
```

2. 运用`./scripts/tracetool.py`工具自动生成`trace.h`和`trace.c`文件

```
Usage: ../scripts/tracetool.py --format=<format> --backends=<backends>
[<options>]
Backends:
    nop             Tracing disabled.
    dtrace          DTrace/SystemTAP backend.
    ftrace          Ftrace built-in backend.
    log             Stderr built-in backend.
    simple          Simple built-in backend.
    syslog          Syslog built-in backend.
    ust             LTTng User Space Tracing backend.

Formats:
    c               trace/generated-tracers.c
    d               trace/generated-tracers.dtrace (DTrace only).
    h               trace/generated-tracers.h
    log-stap        Generate .stp file that printfs log messages (DTrace
with SystemTAP only).
    simpletrace-stap Generate .stp file that outputs simpletrace binary
traces (DTrace with SystemTAP only).
    stap            Generate .stp file (DTrace with SystemTAP only).
    tcg-h           Generate .h file for TCG code generation.
    tcg-helper-c    Generate trace/generated-helpers.c.
    tcg-helper-h    Generate trace/generated-helpers.h.
    tcg-helper-wrapper-h Generate trace/generated-helpers-wrappers.h.
    ust-events-c    trace/generated-ust.c
    ust-events-h    trace/generated-ust-provider.h

Options:
    --help                  This help message.
    --list-backends         Print list of available backends.
    --check-backends        Check if the given backend is valid.
    --binary <path>         Full path to QEMU binary.
```

```
    --target-type <type>     QEMU emulator target type ('system' or
'user').
    --target-name <name>     QEMU emulator target name.
    --group <name>           Name of the event group
    # 这个和 #define 宏定义有关
    #ifndef TRACE_WMS_GENERATED_TRACERS_H
    --probe-prefix <prefix>  Prefix for dtrace probe names
                             (default: qemu-<target-type>-<target-
name>).
```

3. 修改源文件

```
   5     /*
   4      * This must happen after any state changes since as soon as an
external
   3      * observer sees this event they might start to prod at the VM
assuming
   2      * it's ready to use.
   1      */
442      // migrate_set_state(&mis->state, MIGRATION_STATUS_ACTIVE,
   1     //                   MIGRATION_STATUS_COMPLETED);
        trace_migrate_set_state_user(&mis->state,
MIGRATION_STATUS_ACTIVE,
   1                         MIGRATION_STATUS_COMPLETED);
        // 修改源代码 上面为修改值
   2     qemu_bh_delete(mis->bh);
   3     migration_incoming_state_destroy();
   4 }
   5
```

4. 重新编译

```shell
  CC      migration/migration.o
  LINK    moxie-softmmu/qemu-system-moxie
  # ....
```

5. qmp命令

```json
{'execute': 'trace-event-set-state', 'arguments': {'name':
'colo_checkpoint_user_log', 'enable': true} }
{'execute': 'human-monitor-command', 'arguments': {'command-line': 'logfile
/home/data/mengsen/log/test.log'}}
```

6. 生成log

```
    15868@1602750016.645000:migrate_set_state_user new state setup
17234@1602750016.646237:migrate_set_state_user new state active
    17716@1602750016.650037:migrate_set_state_user new state active
17716@1602750021.080884:migrate_set_state_user new state completed
    17234@1602750021.247924:migrate_set_state_user new state completed
```

7. 一些细节
    1. 当觉得某项`trace-event`可以再前面增加`disable`，表示取消此项`trace-event`，具体语法如下

       `[disable] <name>(<type1> <arg1>[, <type2> <arg2>] ...) "<format-string>"`
8. 一些教程
    1. https://github.com/qemu/qemu/blob/master/docs/devel/tracing.txt
    2. https://github.com/qemu/qemu/blob/master/trace-events