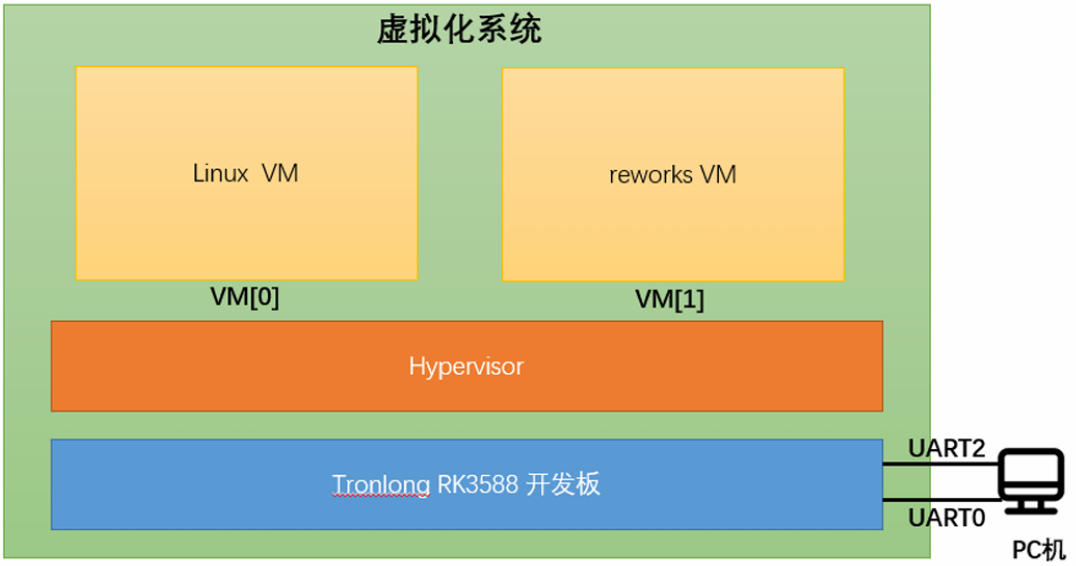

隔离性测试即在 Linux VM 上运行高 cpu、内存开销应用时，不影响 reworks VM 上任务的执行时间，因此需要首先编写一个简单的任务执行时间测试程序 task_running_time

1、 task_running_time 的编写

根据 API 文档，可以使用 clock_gettime()进行计时

```
int clock_gettime ( clockid_t           clock_id,
                    struct timespec *   tp
                  )
```

获取指定时钟的当前值

该接口用于获取指定时钟clock_id的当前值，并存放在参数tp指定的地址中。时钟可以是 系统范围的（即对所有任务可见），或是任务范围的（即度量的时间只对一个任务有意义）。
时钟CLOCK_REALTIME代表系统的实时时钟，对于这个时钟，clock_gettime()的返回值表示从 UTC时间公元1970年1月1日的0时0分0秒以来的时间总量（用秒和纳秒计算）。
时钟CLOCK_MONOTONIC代表系统的单调时钟，对于这个时钟，clock_gettime()的返回值表示 系统自启动以来的时间总量（用秒和纳秒计算）。
时钟CLOCK_THREAD_CPUTIME_ID/CLOCK_PROCESS_CPUTIME_ID代表任务的CPU时间时钟，对于 这个时钟，clock_gettime()的返回值表示任务执行时间的总量。

参数:

clock_id 时钟标识符，其取值范围为[CLOCK_REALTIME, CLOCK_THREAD_CPUTIME_ID]

tp 时钟的当前值

返回:

0 调用成功

-1 出错，并设置错误号指出错误

错误号:

EINVAL 参数tp为非法指针，或者参数clock_id的取值不合法

参见:

clock_settime

示例:

example_clock_getcpuclockid.c、example_clock_gettime.c、example_clock_nanosleep.c、example_clock_settime.c及example_nanosleep.c.

在一个数组写任务开始和结束时进行计时，根据 API 说明，创建

start 和 end 两个 timespec 结构体，并选择 CLOCK_REALTIME

```c
#include "../basic/testing.h"
#include <stdint.h>
#include <stdio.h>
#include <time.h>

//#define DWT_CTRL (*(volatile uint32_t*)0xE0001000)
//#define DWT_CYCCNT (*(volatile uint32_t*)0xE0001004)

uint32_t test_once() {
    struct timespec start, end;

    clock_gettime(CLOCK_REALTIME, &start);
    // 执行任务
    uint8_t buffer[1024];
    for (int i=0; i<10000; i++) {
        for (int j=0; j<1024; j++) {
            buffer[j] = i & 0xFF; // 写操作
        }
    }
    clock_gettime(CLOCK_REALTIME, &end);
    long duration = (end.tv_sec - start.tv_sec) * 1e9 + (end.tv_nsec - start.tv_nsec); //精确到纳秒
    printf("Duration: %ld ns\n", duration);
    return 0;
}

void test_task_running_time() {
    for(int i = 0;i < 100;i++)
    {
        test_once();
    }
}
```

点击 API 文档中的 timespec，进入 timespec 结构体的说明

timespec结构参考

时间值规格结构体类型 更多...

```
#include <types.h>
```

数据成员

time_t	tv_sec 秒
long	tv_nsec 纳秒

可以使用 (end.tv_sec - start.tv_sec) * 1e9 + (end.tv_nsec - start.tv_nsec); 来使计时精确到纳秒。

2、隔离性测试

①、基于《rk3588 搭载虚拟化测试环境教程》，将 task_running_time.c 烧录至板卡中，在 linux_vm 空闲时直接运行

②、在 linux VM 上执行高 cpu、memory 开销的命令，在 reworks VM 再次执行一次 task_running_time.c

③、比较两次执行的时间消耗，如果结果相似则表明有良好的隔离性