作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师
https://github.com/drink-cat/Book_Program-Principles

内存屏障的定义

文件 linux-5.6.3/include/linux/compiler-gcc.h。
编译期屏障，使用 memory 标识符。

```
/* Optimization barrier */

/* The "volatile" is due to gcc bugs */
#define barrier() __asm__ __volatile__("": : :"memory")
```

文件 linux-5.6.3/include/asm-generic/barrier.h。

```
#ifdef CONFIG_SMP

#ifndef smp_mb
#define smp_mb()    __smp_mb()
#endif

#ifndef smp_rmb
#define smp_rmb()   __smp_rmb()
#endif

#ifndef smp_wmb
#define smp_wmb()   __smp_wmb()
#endif
```

文件 linux-5.6.3/arch/x86/include/asm/barrier.h。
编译期屏障、运行期屏障。

```
#ifdef CONFIG_X86_32
#define mb() asm volatile(ALTERNATIVE("lock; addl $0,-4(%%esp)", "mfence", \
                      X86_FEATURE_XMM2) ::: "memory", "cc")
#define rmb() asm volatile(ALTERNATIVE("lock; addl $0,-4(%%esp)", "lfence", \
                       X86_FEATURE_XMM2) ::: "memory", "cc")
#define wmb() asm volatile(ALTERNATIVE("lock; addl $0,-4(%%esp)", "sfence", \
                       X86_FEATURE_XMM2) ::: "memory", "cc")
#else
#define mb()    asm volatile("mfence":::"memory")
#define rmb()   asm volatile("lfence":::"memory")
#define wmb()   asm volatile("sfence" ::: "memory")
#endif

#define dma_rmb()   barrier()
#define dma_wmb()   barrier()

#ifdef CONFIG_X86_32
```

```
#define __smp_mb()  asm volatile("lock; addl $0,-4(%%esp)" ::: "memory", "cc")
#else
#define __smp_mb()  asm volatile("lock; addl $0,-4(%%rsp)" ::: "memory", "cc")
#endif
#define __smp_rmb() dma_rmb()
#define __smp_wmb() barrier()
```

内存屏障的使用举例

文件 linux-5.6.3/kernel/events/ring_buffer.c。

```
static void perf_output_put_handle(struct perf_output_handle *handle)
{
    struct perf_buffer *rb = handle->rb;
    unsigned long head;
    unsigned int nest;

    /*
     * If this isn't the outermost nesting, we don't have to update
     * @rb->user_page->data_head.
     */
    nest = READ_ONCE(rb->nest);
    if (nest > 1) {
        WRITE_ONCE(rb->nest, nest - 1);
        goto out;
    }

again:
    /*
     * In order to avoid publishing a head value that goes backwards,
     * we must ensure the load of @rb->head happens after we've
     * incremented @rb->nest.
     *
     * Otherwise we can observe a @rb->head value before one published
     * by an IRQ/NMI happening between the load and the increment.
     */
    barrier();
    head = local_read(&rb->head);
```

文件 linux-5.6.3/kernel/exit.c。
使用 smp_mb()保证 2 个函数的变量的相互依赖的顺序。

```
void rcuwait_wake_up(struct rcuwait *w)
{
    struct task_struct *task;

    rcu_read_lock();
```

```
    /*
     * Order condition vs @task, such that everything prior to the load
     * of @task is visible. This is the condition as to why the user called
     * rcuwait_trywake() in the first place. Pairs with set_current_state()
     * barrier (A) in rcuwait_wait_event().
     *
     *    WAIT                WAKE
     *    [S] tsk = current   [S] cond = true
     *        MB (A)          MB (B)
     *    [L] cond        [L] tsk
     */
    smp_mb(); /* (B) */

    task = rcu_dereference(w->task);
    if (task)
        wake_up_process(task);
    rcu_read_unlock();
}
```

文件 linux-5.6.3/drivers/xen/time.c。
使用 rmb()读到最新值。

```
static void xen_get_runstate_snapshot_cpu_delta(
                  struct vcpu_runstate_info *res, unsigned int cpu)
{
    u64 state_time;
    struct vcpu_runstate_info *state;

    BUG_ON(preemptible());

    state = per_cpu_ptr(&xen_runstate, cpu);

    do {
        state_time = get64(&state->state_entry_time);
        rmb();  /* Hypervisor might update data. */
        *res = READ_ONCE(*state);
        rmb();  /* Hypervisor might update data. */
    } while (get64(&state->state_entry_time) != state_time ||
         (state_time & XEN_RUNSTATE_UPDATE));
}
```