Kernel 里面对抢占的关闭和打开，是依赖 preempt_count 这个变量的，而不是某个开关。
preempt_count 是个 u64 变量，属于 struct thread_info 结构体。
以 arch\arm64\include\asm\thread_info.h 为例：

```
/*
 * low level task data that entry.S needs immediate access to.
 */
struct thread_info {
	unsigned long		flags;		/* low level flags */
	mm_segment_t		addr_limit;	/* address limit */
#ifdef CONFIG_ARM64_SW_TTBR0_PAN
	u64			ttbr0;		/* saved TTBR0_EL1 */
#endif
	union {
		u64		preempt_count;	/* 0 => preemptible, <0 => bug */
		struct {
#ifdef CONFIG_CPU_BIG_ENDIAN
			u32	need_resched;
			u32	count;
#else
			u32	count;
			u32	need_resched;
#endif
		} preempt;
	};
};
```

=0 可以被抢占，>0 不能被抢占。
读取这个值可以用函数：

```
static inline int preempt_count(void)
{
	return READ_ONCE(current_thread_info()->preempt.count);
}
```

不同的场景，抢占的值是不一样的。

reserved bits	bit21	bit20	bit19-bit16	bit15-bit8	bit7-bit0
	PREEMPT_ACTIVE	NMI	HARDIRQ	SOFTIRQ	PREEMPT

```
/*
 * PREEMPT_MASK:        0x000000ff
 * SOFTIRQ_MASK:        0x0000ff00
 * HARDIRQ_MASK:        0x000f0000
 *     NMI_MASK:        0x00100000
 * PREEMPT_ACTIVE:      0x00200000
 */
#define PREEMPT_BITS    8
#define SOFTIRQ_BITS    8
#define HARDIRQ_BITS    4
#define NMI_BITS        1
```

普通场景(PREEMPT_MASK)打开和关闭抢占：
arch\arm64\include\asm\preempt.h

```
#define preempt_enable() \
do { \
	barrier(); \
	if (unlikely(preempt_count_dec_and_test())) \
		__preempt_schedule(); \
} while (0)
```

```
static __always_inline bool __preempt_count_dec_and_test(void)
{
	/*
	 * Because of load-store architectures cannot do per-cpu atomic
	 * operations; we cannot use PREEMPT_NEED_RESCHED because it might get
	 * lost.
	 */
	return !--*preempt_count_ptr() && tif_need_resched();
}
```

```
static __always_inline volatile int *preempt_count_ptr(void)
{
	return &current_thread_info()->preempt_count;
}
```

可以看到此处只是 preempt_count++或者--
软中断场景（SOFTIRQ_MASK）打开和关闭抢占：

Kernel\softirq.c

```
/*
 * Special-case - softirqs can safely be enabled by __do_softirq(),
 * without processing still-pending softirqs:
 */
void _local_bh_enable(void)
{
	WARN_ON_ONCE(in_irq());
	__local_bh_enable(SOFTIRQ_DISABLE_OFFSET);
}

static void __local_bh_enable(unsigned int cnt)
{
	lockdep_assert_irqs_disabled();

	if (preempt_count() == cnt)
		trace_preempt_on(CALLER_ADDR0, get_lock_parent_ip());

	if (softirq_count() == (cnt & SOFTIRQ_MASK))
		trace_softirqs_on(_RET_IP_);

	__preempt_count_sub(cnt);
}

static __always_inline void __preempt_count_sub(int val)
{
	*preempt_count_ptr() -= val;
}
```

可以看到此处增减的幅度是 SOFTIRQ_DISABLE_OFFSET