current 宏的原理，就是通过拿到当前栈指针 SP，清掉 8k 内核栈的偏移，拿到 thread_info 结构体的入口地址，然后从 thread_info 指向 task 成员，得到 task_struct 结构体。

参考：
内核 current 宏解析.pdf
How does the ''current'' macro helps you access your process-related record.pdf

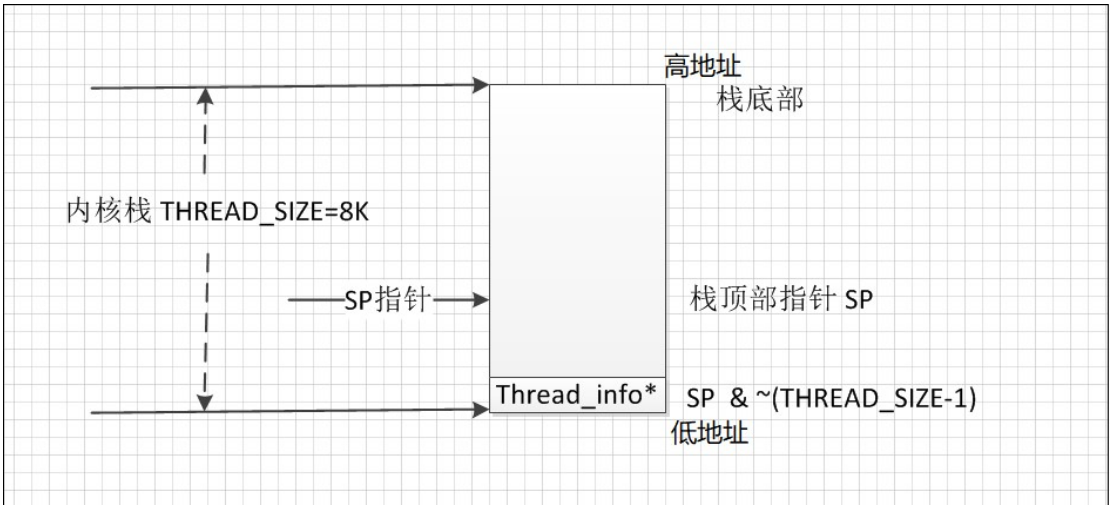
首先虽然 task_struct 是 arch 无关的通用结构体。
thread_info 却是 arch 相关的结构体，且内部有指向 task_struct 的成员
以 arm32 的 arch/arm/include/asm/thread_info.h 为例：

```
struct thread_info {
        unsigned long           flags;          /* low level flags */
        int                     preempt_count;  /* 0 => preemptable, <0 => bug */
        mm_segment_t            addr_limit;     /* address limit */
        struct task_struct      *task;          /* main task structure */
        __u32                   cpu;            /* cpu */
        __u32                   cpu_domain;     /* cpu domain */
#ifdef CONFIG_STACKPROTECTOR_PER_TASK
        unsigned long           stack_canary;
#endif
        struct cpu_context_save         cpu_context;    /* cpu context */
        __u32                   syscall;        /* syscall number */
        __u8                    used_cp[16];    /* thread used copro */
        unsigned long           tp_value[2];    /* TLS registers */
#ifdef CONFIG_CRUNCH
        struct crunch_state     crunchstate;
#endif
        union fp_state          fpstate __attribute__((aligned(8)));
        union vfp_state         vfpstate;
#ifdef CONFIG_ARM_THUMBEE
        unsigned long           thumbee_state;  /* ThumbEE Handler Base register */
#endif
};
```

thread_info 和线程的栈一起，被放在一个叫 thread_union 的共同体里面：

```
union thread_union {
#ifndef CONFIG_ARCH_TASK_STRUCT_ON_STACK
        struct task_struct task;
#endif
#ifndef CONFIG_THREAD_INFO_IN_TASK
        struct thread_info thread_info;
#endif
        unsigned long stack[THREAD_SIZE/sizeof(long)];
};
```

所以 thread_info 和栈 stack 是占用同一块空间的。
这就有了这张图：



如果已知 SP 指针，因为 stack 默认是从高地址往低地址生长的，所以图中栈底部反而是在高地址。

arm32 有如下定义：

```
#define THREAD_SIZE_ORDER       1
#define THREAD_SIZE             (PAGE_SIZE << THREAD_SIZE_ORDER)
```

PAGE_SIZE 是 4096，所以 THREAD_SIZE 是 8192

所以做法就是把 SP 的往 8192 对齐的低地址部分全部清掉。

8192=0x2000=0010000000000000

8192-1=0x1FFF=0001111111111111

再做&~的操作，等于将 SP 的低 12bit 全部清零，得到的就是图中的低地址位置，即 thread_info 结构体的起始位置。再用 thread_info->task 即可得到 task_struct。

所以代码实现如下：

```
#define get_current() (current_thread_info()->task)
#define current get_current()

static inline struct thread_info *current_thread_info(void)
{
        return (struct thread_info *)
                (current_stack_pointer & ~(THREAD_SIZE - 1));
}
```