

专题8-Linux系统调用

一、作用介绍

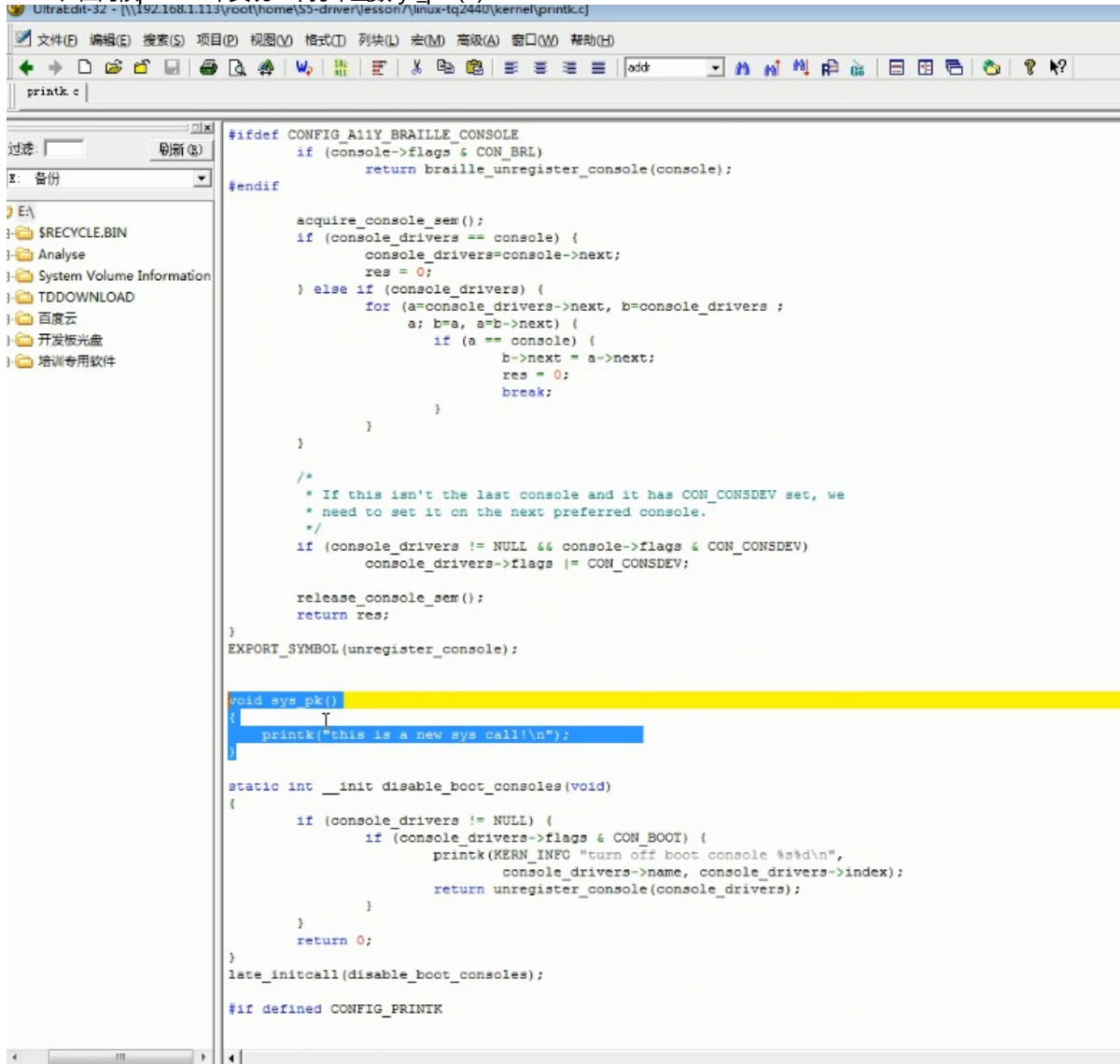
实现体在内核空间，提供给用户空间应用使用。

二、工作流程分析

系统调用编号（ARM通常储存在R7）

三、实现新的系统调用

3.1、在内核printk.c中实现一个打印函数sys_pk（）：



```
#ifndef CONFIG_A11Y_BRAILLE_CONSOLE
    if (console->flags & CON_BRL)
        return braille_unregister_console(console);
#endif

acquire_console_sem();
if (console_drivers == console) {
    console_drivers=console->next;
    res = 0;
} else if (console_drivers) {
    for (a=console_drivers->next, b=console_drivers;
         a; b=a, a=b->next) {
        if (a == console) {
            b->next = a->next;
            res = 0;
            break;
        }
    }
}

/*
 * If this isn't the last console and it has CON_CONSDEV set, we
 * need to set it on the next preferred console.
 */
if (console_drivers != NULL && console->flags & CON_CONSDEV)
    console_drivers->flags |= CON_CONSDEV;

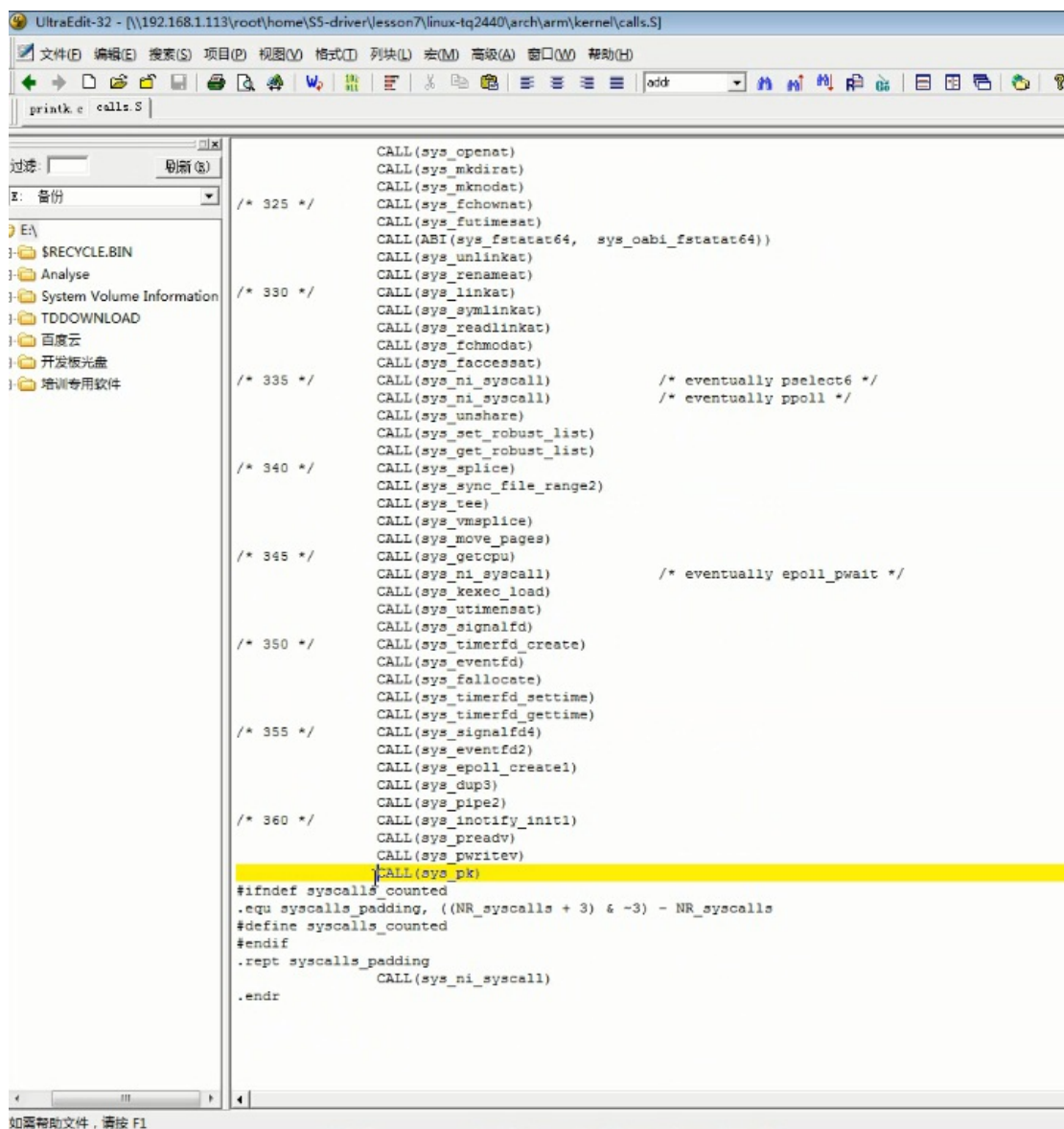
release_console_sem();
return res;
}
EXPORT_SYMBOL(unregister_console);

void sys_pk()
{
    printk("this is a new sys call!\n");
}

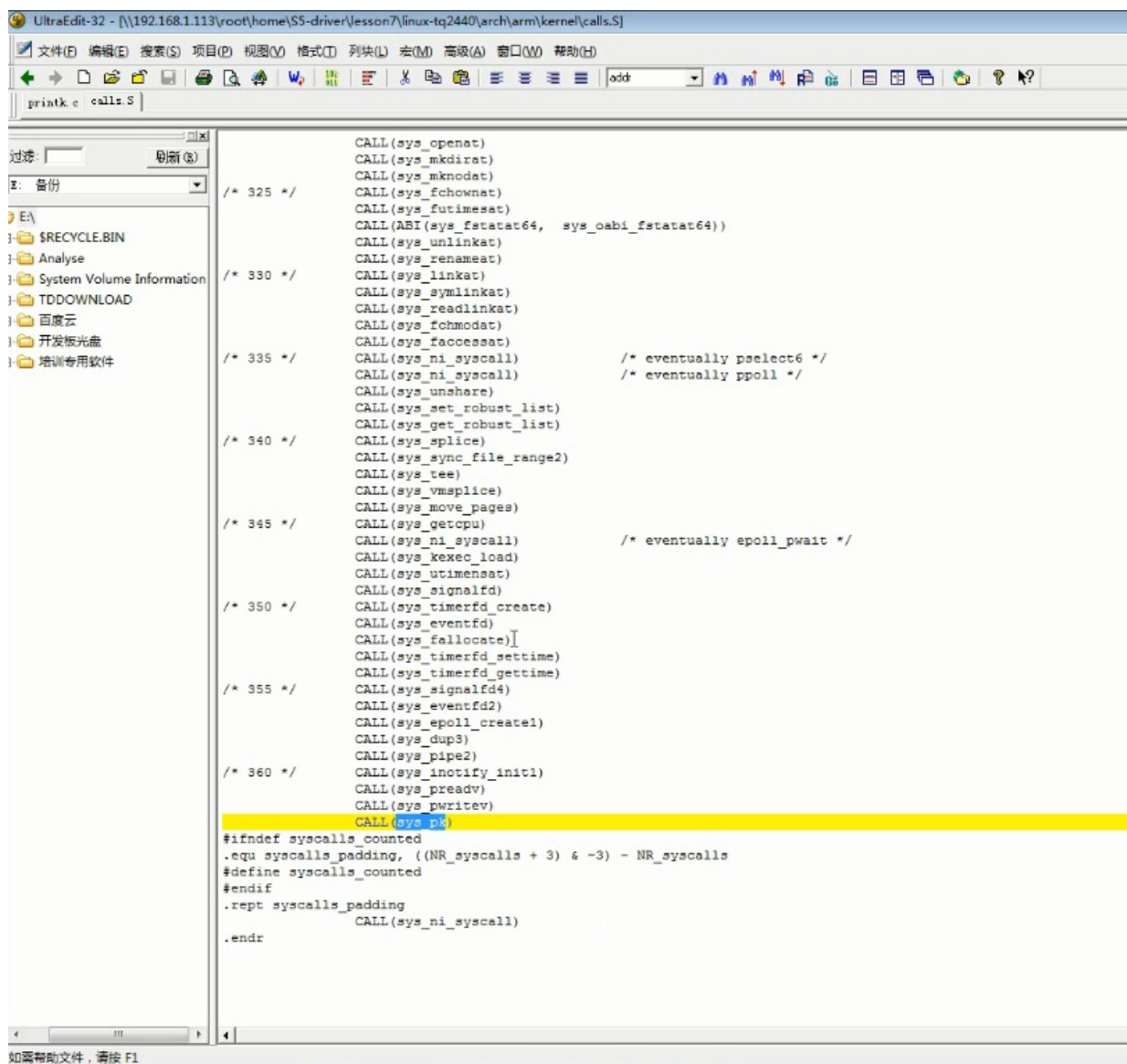
static int __init disable_boot_consoles(void)
{
    if (console_drivers != NULL) {
        if (console_drivers->flags & CON_BOOT) {
            printk(KERN_INFO "turn off boot console %s%d\n",
                   console_drivers->name, console_drivers->index);
            return unregister_console(console_drivers);
        }
    }
    return 0;
}
late_initcall(disable_boot_consoles);

#if defined CONFIG_PRINTK
```

3.2、在calls.S中最后加入系统调用，并记下编号



3.3、在unistd.h (包含系统调用编号) 中对应系统编号添加系统调用



3.4、重新编译内核

3.5、用户空间使用函数调用系统函数

```

void pk()
{
    __asm__(
        "ldr r7,=363\n"
        "swi\n"
        :
        :
        : "memory");
}

void main()
{
    pk();
}

```