Linux内核编程03期：系统调用

文档配套视频地址：https://wanglitao.taobao.com
专注嵌入式精品教程：www.zhaixue.cc

kill()系统调用

系统调用号

```
arch/arm/include/generated/uapi/asm/unistd-common.h

#define __NR_kill (__NR_SYSCALL_BASE + 37)
```

系统调用函数声明

```
include/linux/syscalls.h

asmlinkage:GCC扩展，表示读取的参数来自栈中，而非寄存器
/* kernel/signal.c */
asmlinkage long sys_restart_syscall(void);
asmlinkage long sys_kill(pid_t pid, int sig);
asmlinkage long sys_tkill(pid_t pid, int sig);
```

系统调用函数实现

```
kernel/signal.c   :

SYSCALL_DEFINE2(kill, pid_t, pid, int, sig)
{
    struct kernel_siginfo info;
    prepare_kill_siginfo(sig, &info);
    return kill_something_info(sig, &info, pid);
}

展开后相当于:
asmlinkage long sys_kill(pid_t pid, int sig)
```

```
arch/arm/kernel/entry-common.S      :  保护现场，获取系统调用号
ENTRY(vector_swi)
addne scno, r7, #__NR_SYSCALL_BASE□@ put OS number in
ldr tbl, sys_call_table
...
invoke_syscall tbl, scno, r10, __ret_fast_syscall
   add  r1, sp, #S_OFF
2: cmp  scno, #(__ARM_NR_BASE - □__NR_SYSCALL_BASE)
   eor  r0, scno, #__NR_SYSCALL_BASE @ put OS number back
   bcs  arm_syscall
   mov  why, #0□□ @ no longer a real syscall
   b   sys_ni_syscall□□ @ not private func
...
9001:
  sub lr, saved_pc, #4
  str lr, [sp, #S_PC]
  get_thread_info tsk
  b ret_fast_syscall   回到用户态kill，继续执行用户态代码
ENDPROC(vector_swi)

syscall_table_start sys_call_table
  #define COMPAT(nr, native, compat) syscall nr, native
  #ifdef CONFIG_AEABI
    #include <calls-eabi.S>
  #else
    #include <calls-oabi.S>
  #endif
  #undef COMPAT
syscall_table_end sys_call_table

#define NATIVE(nr, func) syscall nr, func
```

```
arch/arm/include/generated/calls-eabi.S       :
NATIVE(0, sys_restart_syscall)
NATIVE(1, sys_exit)
NATIVE(2, sys_fork)
NATIVE(3, sys_read)
NATIVE(4, sys_write)
NATIVE(5, sys_open)
NATIVE(6, sys_close)
NATIVE(8, sys_creat)
NATIVE(9, sys_link)
NATIVE(10, sys_unlink)
NATIVE(11, sys_execve)
NATIVE(12, sys_chdir)
NATIVE(14, sys_mknod)
NATIVE(15, sys_chmod)
NATIVE(16, sys_lchown16)
NATIVE(19, sys_lseek)
NATIVE(20, sys_getpid)
NATIVE(21, sys_mount)
NATIVE(23, sys_setuid16)
NATIVE(24, sys_getuid16)
NATIVE(26, sys_ptrace)
NATIVE(29, sys_pause)
NATIVE(33, sys_access)
NATIVE(34, sys_nice)
NATIVE(36, sys_sync)
NATIVE(37, sys_kill)
NATIVE(38, sys_rename)
NATIVE(39, sys_mkdir)
其实就是定义一个函数入口指针  .long sys_kill
```

```
tools/include/nolibc/nolibc.h:
#define my_syscall2(num, arg1, arg2)                              \
({                                                                \
□register long _num asm("r7") = (num);          \
□register long _arg1 asm("r0") = (long)(arg1); \
□register long _arg2 asm("r1") = (long)(arg2); \
□□□□□\
□asm volatile (                                 \
□□"svc #0\n"                         \
□□: "=r"(_arg1)                      \
□□: "r"(_arg1), "r"(_arg2),          \
□□  "r"(_num)                        \
□□: "memory", "cc", "lr"             \
□);                                            \
□_arg1;                                        \
})

static  __attribute__((unused))
int sys_kill(pid_t pid, int signal)
{
□return my_syscall2(__NR_kill, pid, signal);
}

static __attribute__((unused))
int kill(pid_t pid, int signal)
{
□int ret = sys_kill(pid, signal);

□if (ret < 0) {
□□SET_ERRNO(-ret);
□□ret = -1;
□}
□return ret;
}
```

```
/usr/arm-linux-gnueabi/lib/libc.a:

00000000 <__kill>:
   0:□e52d7004 □push□{r7}□; (str r7, [sp, #-4]!)
   4:□e3a07025 □mov□r7, #37□; 0x25
   8:□ef000000 □svc□0x00000000
   c:□e49d7004 □pop□{r7}□; (ldr r7, [sp], #4)
  10:□e3700a01 □cmn□r0, #4096□; 0x1000
  14:□312fff1e □bxcc□lr
  18:□eafffffe □b□0 <__syscall_error>
```

SVC指令，即以前的SWI指令，软中断

```
/usr/arm-linux-gnueabi/lib/libc-2.31.so:

000dad70 <syscall@@GLIBC_2.4>:
   dad70:□e1a0c00d □mov□ip, sp
   dad74:□e92d00f0 □push□{r4, r5, r6, r7}
   dad78:□e1a07000 □mov□r7, r0
   dad7c:□e1a00001 □mov□r0, r1
   dad80:□e1a01002 □mov□r1, r2
   dad84:□e1a02003 □mov□r2, r3
   dad88:□e89c0078 □ldm□ip, {r3, r4, r5, r6}
   dad8c:□ef000000 □svc□0x00000000
   dad90:□e8bd00f0 □pop□{r4, r5, r6, r7}
   dad94:□e3700a01 □cmn□r0, #4096□; 0x1000
   dad98:□312fff1e □bxcc□lr
   dad9c:□eafcf2c3 □b□178b0 <__libc_start_main@@GLIBC_2.4+0x278>
```