ISCAS NIST

循序渐进，学习开发一个 RISC-V 上的操作系统

第 16 章 系统调用

汪辰

2021/4

本章内容

- 系统模式：用户态和内核态
- 系统模式的切换
- 系统调用的执行流程
- 系统调用的传参
- 系统调用的封装

参考

- 【参考 1】：The RISC-V Instruction Set Manual，Volume I: Unprivileged ISA，Document Version 20191213
- 【参考 2】: The RISC-V Instruction Set Manual，Volume II: Privileged Architecture，Document Version 20190608-Priv-MSU-Ratified

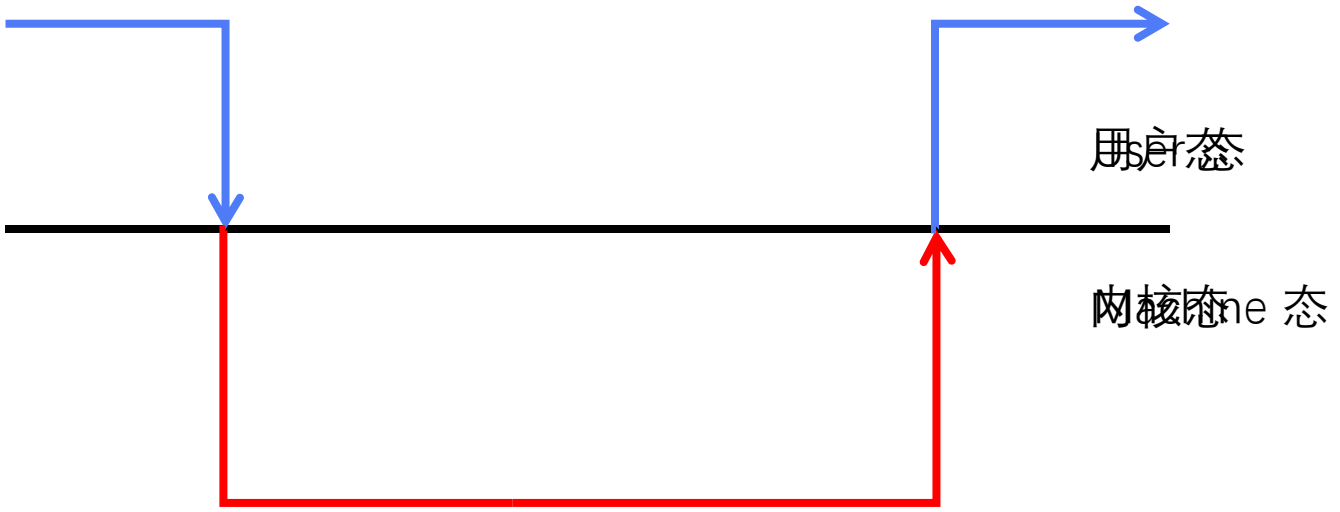
本章内容

- **系统模式：用户态和内核态**
- **系统模式的切换**
- **系统调用的执行流程**
- **系统调用的传参**
- **系统调用的封装**

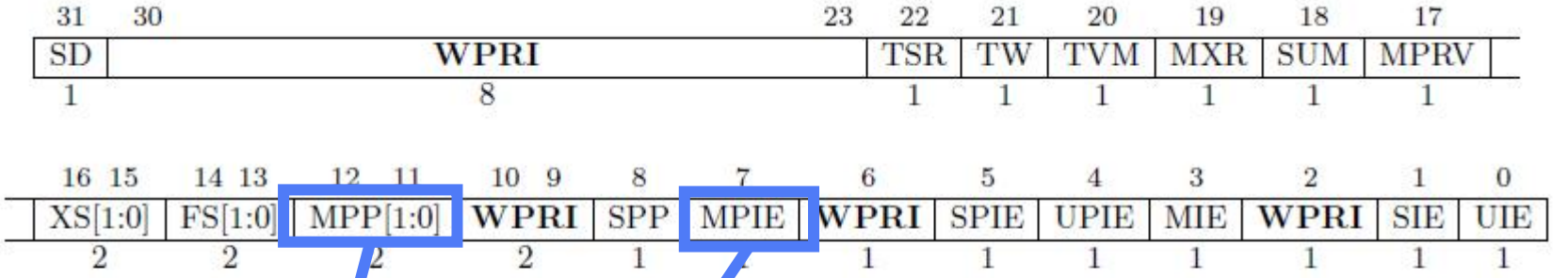
系统模式：用户态和内核态

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

【参考 2】 Table 1.2: Supported combinations of privilege modes.



系统模式：用户态和内核态



【参考 2】Figure 3.6: Machine-mode status register (mstatus) for RV32.

mstatus 上电后默认为 0

code/os/08-preemptive/start.S

```
_start
        ......
        li      t0, 3 << 11 | 1 << 7
        csrr    a1, mstatus
        or      t0, t0, a1
        csrw    mstatus, t0

        j       start_kernel
```

code/os/08-preemptive/kernel.c

```
void start_kernel(void)
{
        ......
        schedule();

        uart_puts("Would not go here!\n");
        while (1) {}; // stop here!
}
```

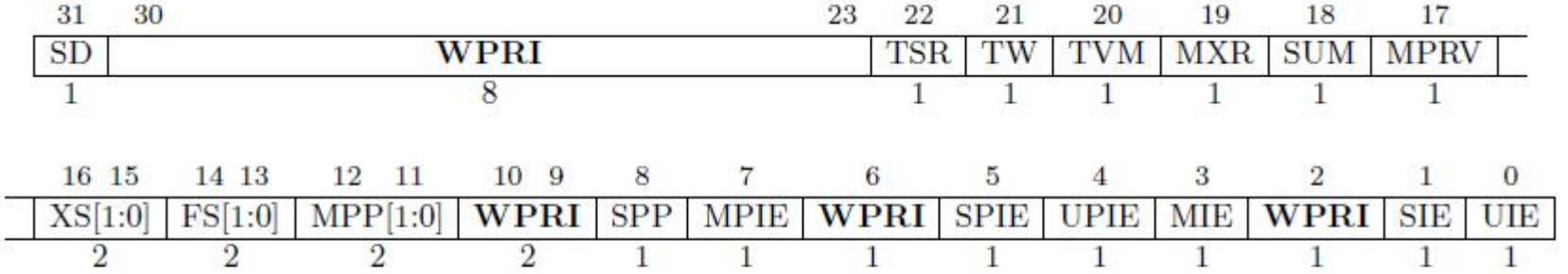
code/os/08-preemptive/sched.c

```
void schedule()
{
        ......
        switch_to(next);
}
```

code/os/08-preemptive/entry.S

```
switch_to:
        ......
        mret
```

系统模式：用户态和内核态



31	30 … 23	22	21	20	19	18	17
SD	WPRI	TSR	TW	TVM	MXR	SUM	MPRV
1	8	1	1	1	1	1	1

16 15	14 13	12 11	10 9	8	7	6	5	4	3	2	1	0
XS[1:0]	FS[1:0]	MPP[1:0]	WPRI	SPP	MPIE	WPRI	SPIE	UPIE	MIE	WPRI	SIE	UIE
2	2	2	2	1	1	1	1	1	1	1	1	1

【参考 2】 Figure 3.6: Machine-mode status register (mstatus) for RV32.

code/os/08-preemptive/start.S

```
_start:
        ......
        li      t0, 3 << 11 | 1 << 7
        csrr    a1, mstatus
        or      t0, t0, a1
        csrw    mstatus, t0

        j       start_kernel
```

code/os/11-syscall/start.S

```
_start:
        ......
        li      t0, 1 << 7
        csrr    a1, mstatus
        or      t0, t0, a1
        csrw    mstatus, t0

        j       start_kernel
```

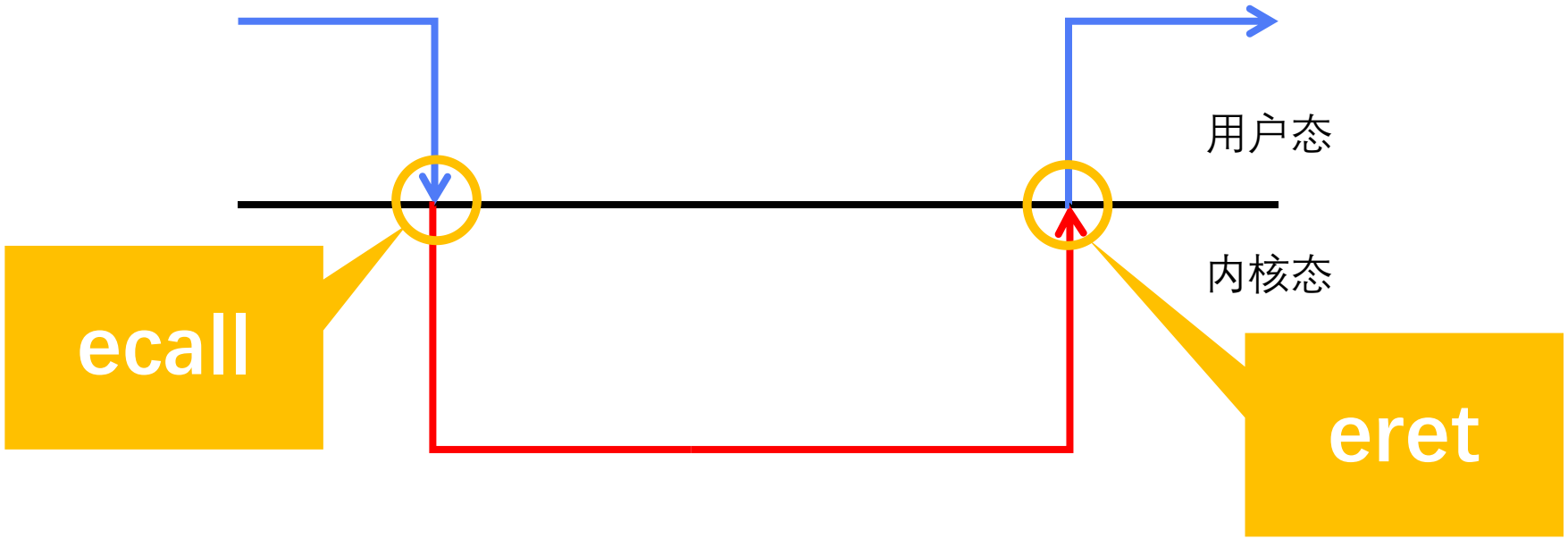
本章内容

- 系统模式：用户态和内核态
- **系统模式的切换**
- 系统调用的执行流程
- 系统调用的传参
- 系统调用的封装

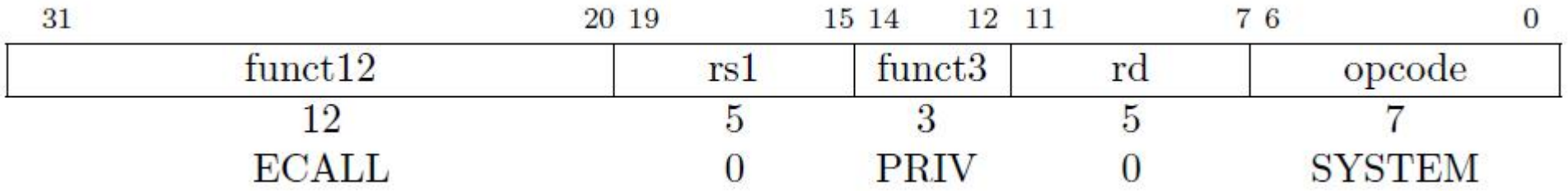
系统模式的切换

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

【参考 2】Table 1.2: Supported combinations of privilege modes.



系统模式的切换



【参考 2】3.2.1 Environment Call and Breakpoint

- ECALL 命令用于主动触发异常
- 根据调用 ECALL 的权限级别产生不同的 exception code
- 异常产生时 epc 寄存器的值存放的是 ECALL 指令本身的地址。

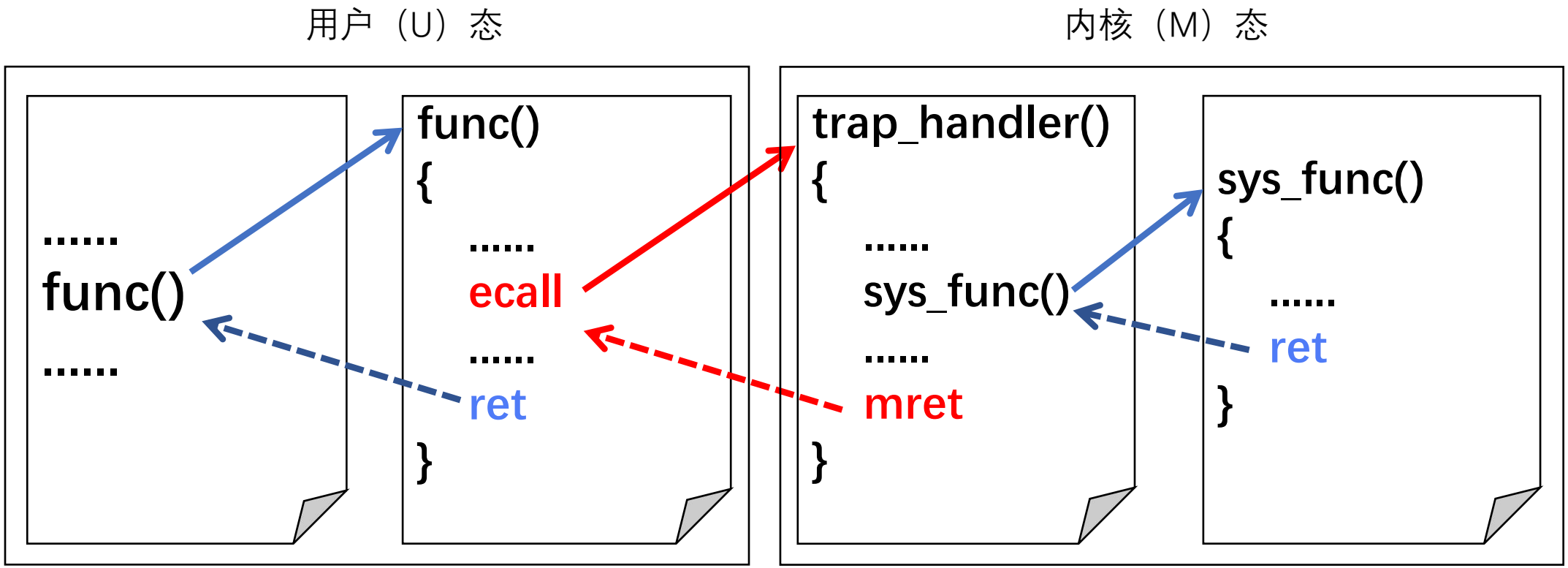
Interrupt	Exception Code	Description
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	*Reserved*
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved for future standard use*
0	15	Store/AMO page fault
0	16–23	*Reserved for future standard use*
0	24–31	*Reserved for custom use*
0	32–47	*Reserved for future standard use*
0	48–63	*Reserved for custom use*
0	≥64	*Reserved for future standard use*

【参考 2】 Table 3.6: Machine cause register (`mcause`) values after trap.

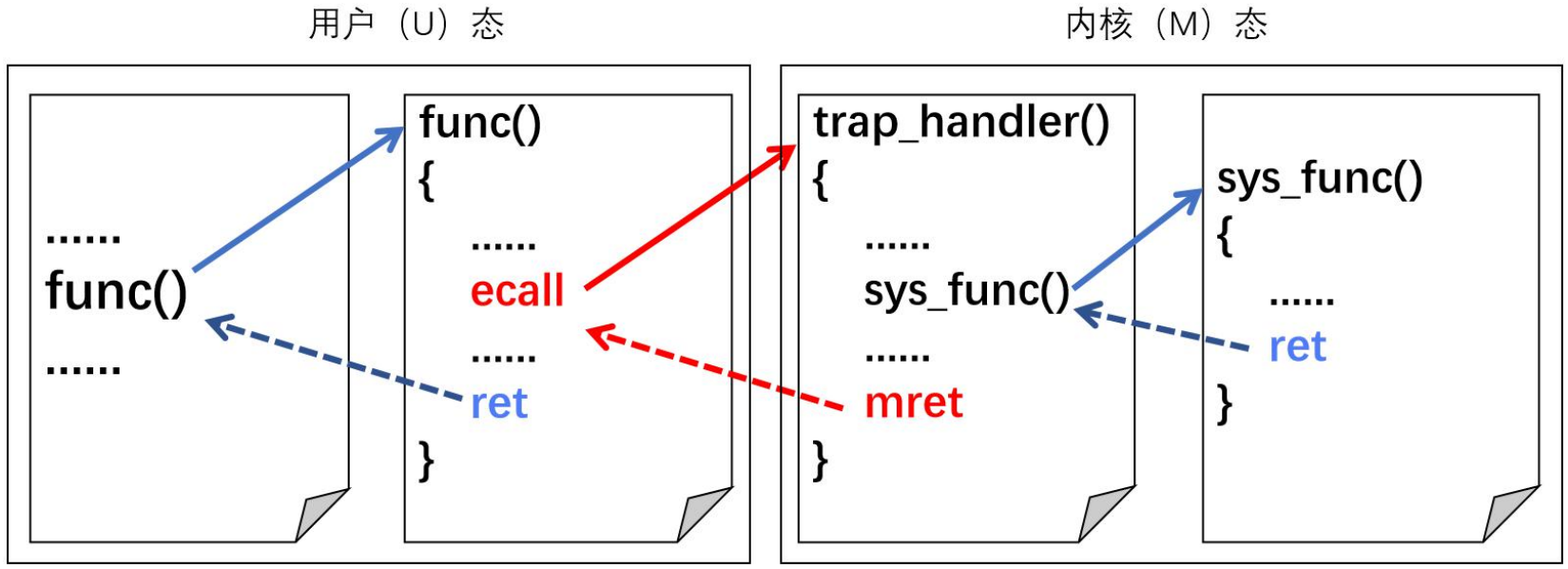
本章内容

- 系统模式：用户态和内核态
- 系统模式的切换
- **系统调用的执行流程**
- 系统调用的传参
- 系统调用的封装

系统调用的执行流程


用户（U）态
内核（M）态
.....
func()
.....
func()
{
.....
ecall
.....
ret
}
trap_handler()
{
.....
sys_func()
.....
mret
}
sys_func()
{
.....
ret
}

系统调用的执行流程



code\os\11-syscall\usys.S

```
.global gethid
gethid:
        ......
        ecall
        ret
```

code/os/11-syscall/entry.S

```
trap_vector:
        ......
        call    trap_handler
        ......
        mret
```

code/os/11-syscall/trap.c

```
reg_t trap_handler(......)
{
        ......
        if (cause & 0x80000000) {
                ......
        } else {
                switch (cause_code) {
                case 8:
                        do_syscall(cxt);
                        return_pc += 4;
                        break;
                ......
        }
        return return_pc;
}
```

code/os/11-syscall/syscall.c

```
void do_syscall(......)
{
        ......
        sys_gethid(......);
        ......
}
```

本章内容

- 系统模式：用户态和内核态
- 系统模式的切换
- 系统调用的执行流程
- **系统调用的传参**
- 系统调用的封装

系统调用的传参

- **系统调用作为操作系统的对外接口，由操作系统的实现负责定义。参考 Linux 的系统调用，RVOS 定义系统调用的传参规则如下：**
  - 系统调用号放在 a7 中
  - 系统调用参数使用 a0 ~ a5
  - 返回值使用 a0

系统调用的传参

code/os/11-syscall/user.c

```
void user_task0(void)
{
        ......
        unsigned int hid = -1;
        int ret = -1;
        ret = gethid(&hid);
        ......
}
```

code/os/11-syscall/syscall.h

```
// System call numbers
#define SYS_gethid      1
```

code/os/11-syscall/usys.S

```
#include "syscall.h"
.global gethid
gethid:
        li a7, SYS_gethid
        ecall

        ret
```

code/os/11-syscall/entry.S

```
trap_vector:
        ......
        csrr    a0, mepc
        csrr    a1, mcause
        csrr    a2, mscratch
        call    trap_handler
        ......
        mret
```

reg_save

context pointer

code/os/11-syscall/trap.c

```
reg_t trap_handler(...,
                   struct context *cxt)
{
        ......
        do_syscall(cxt);
        ......
}
```

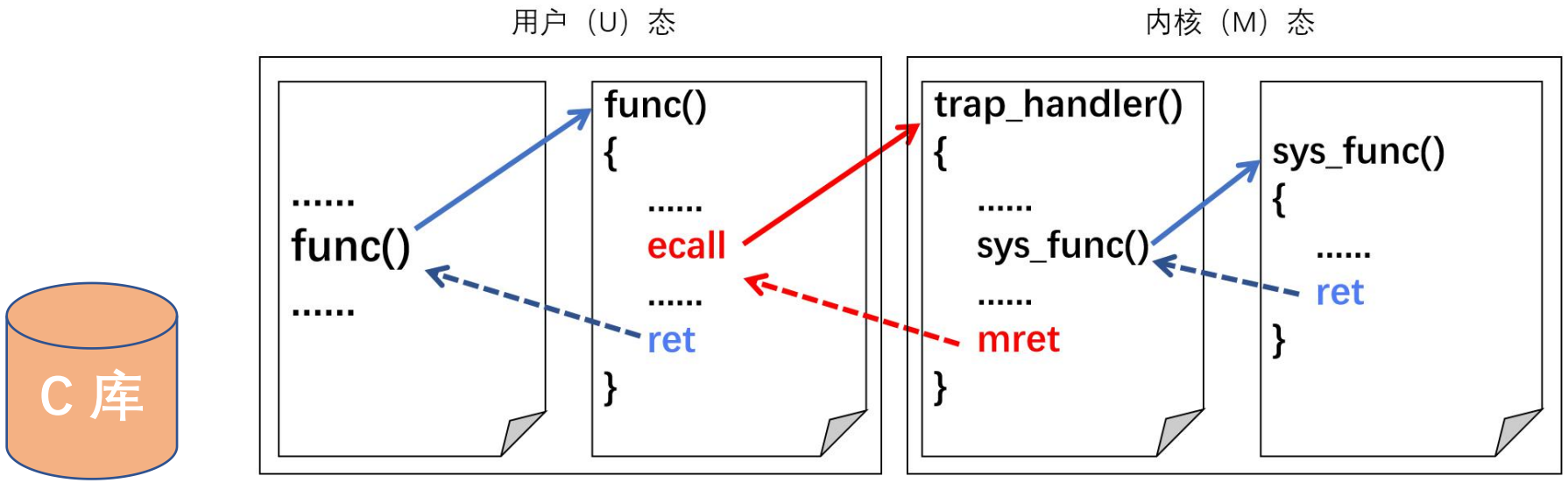
code/os/11-syscall/syscall.c

```
void do_syscall(struct context *cxt)
{
        uint32_t syscall_num = cxt->a7;
        switch (syscall_num) {
        case SYS_gethid:
                cxt->a0 = sys_gethid((unsigned int *)(cxt->a0));
                break;
        default:
                printf("Unknown syscall no: %d\n", syscall_num);
                cxt->a0 = -1;
        }
        return;
}
```

本章内容

- 系统模式：用户态和内核态
- 系统模式的切换
- 系统调用的执行流程
- 系统调用的传参
- **系统调用的封装**

系统调用的封装



code/os/11-syscall/syscall.h

```
// System call number// System call numbers          l numbers
#define SYS_gethid    #define SYS_gethid       1       ethid       1
```

code/os/11-syscall/usys.S

```
#include "syscall.h"
.global gethid
gethid:
        li a7, SYS_gethid
        ecall

        ret
```

code/os/11-syscall/syscall.c

```
int sys_gethid(unsigned int *ptr_hid)
{
        printf("--> sys_gethid, arg0 = 0x%x\n", ptr_hid);
        if (ptr_hid == NULL) {
                return -1;
        } else {
                *ptr_hid = r_mhartid();
                return 0;
        }
}
```

练习 16-1

谢 谢

欢迎交流合作