

浙江大学操作系统实验报告

实验名称: Lab 1: RV64 内核引导 与 时钟中断处理

电子邮件地址: jjhuang535@outlook.com

手机: 18972039196

实验地点: 曹西503

实验日期: 2023.10.3

学号: 3210105944

姓名: 黄锦骏

一、实验目的

- 学习 RISC-V 汇编, 编写 head.S 实现跳转到内核运行的第一个 C 函数。
- 学习 OpenSBI, 理解 OpenSBI 在实验中所起到的作用, 并调用 OpenSBI 提供的接口完成字符的输出。
- 学习 Makefile 相关知识, 补充项目中的 Makefile 文件, 来完成对整个工程的管理。
- 学习 RISC-V 的 trap 处理相关寄存器与指令, 完成对 trap 处理的初始化。
- 理解 CPU 上下文切换机制, 并正确实现上下文切换功能。
- 编写 trap 处理函数, 完成对特定 trap 的处理。
- 调用 OpenSBI 提供的接口, 完成对时钟中断事件的设置。

二、实验过程

2.1 编写Head.S

将栈顶指针指向栈, 并且跳转到start_kernel

```
1  .extern start_kernel
2
3  .section .text.entry
4  .globl _start
5  _start:
6      la sp, boot_stack_top
7      jal start_kernel
8  .section .bss.stack
9  .globl boot_stack
10 boot_stack:
11     .space 4096 # <-- change to your stack size
12
13     .globl boot_stack_top
14 boot_stack_top:
15
```

2.2 编写Makefile

通过阅读工程中的 Makefile 文件，我补全了lib/Makefile文件，该文件内容如下：

```
1  C_SRC      = $(sort $(wildcard *.c))
2  OBJ        = $(patsubst %.c,%.o,$(C_SRC))
3
4  all:$(OBJ)
5
6  %.o:%.c
7      ${GCC}  ${CFLAG} -c $<
8
9  clean:
10     $(shell rm *.o 2>/dev/null)
11
```

其中，我们定义了变量C_SRC与OBJ，分别指代路径下所有.c文件与.o文件，以便对所有源文件进行编译，清除所有构建产物。

2.3 编写sbi.c

sbi_ecall 函数中，需要完成以下内容：

1. 将 ext (Extension ID) 放入寄存器 a7 中，fid (Function ID) 放入寄存器 a6 中，将 arg0 ~ arg5 放入寄存器 a0 ~ a5 中。
2. 使用 ecall 指令。ecall 之后系统会进入 M 模式，之后 OpenSBI 会完成相关操作。
3. OpenSBI 的返回结果会存放在寄存器 a0，a1 中，其中 a0 为 error code，a1 为返回值，我们用 sbiret 来接受这两个返回

编写完成的sbi.c文件如下：

```
1  #include "types.h"
2  #include "sbi.h"
3
4
5  struct sbiret sbi_ecall(int ext, int fid, uint64 arg0,
6                          uint64 arg1, uint64 arg2,
7                          uint64 arg3, uint64 arg4,
8                          uint64 arg5)
9  {
10     struct sbiret ret;
11     uint64 error, value;
12     __asm__ volatile (
13         "mv a0, %[arg0]\n"
14         "mv a1, %[arg1]\n"
15         "mv a2, %[arg2]\n"
16         "mv a3, %[arg3]\n"
17         "mv a4, %[arg4]\n"
18         "mv a5, %[arg5]\n"
19         "mv a6, %[fid]\n"
20         "mv a7, %[ext]\n"
21         "ecall\n"
22         "mv %[ret_val], a0\n"
23         "mv %[err_code], a1"
24         : [ret_val]="r"(value), [err_code]="r"(error)
```

```

25         :[arg0]"r"(arg0), [arg1]"r"(arg1), [arg2]"r"(arg2),[arg3]"r"(arg3),
    [arg4]"r"(arg4), [arg5]"r"(arg5), [fid] "r"(fid), [ext] "r" (ext)
26         : "memory"
27     );
28     ret.error = error;
29     ret.value = value;
30     return ret;
31 }
32

```

直接在汇编函数中完成所有的内存移动，并最终将结果保存到Ret中返回即可

2.4 修改defs.h

按照示例编写了defs.h，两个宏的作用是读/写控制状态寄存器

```

1  #ifndef _DEFS_H
2  #define _DEFS_H
3
4  #include "types.h"
5
6  #define csr_read(csr) \
7  ({ \
8      register uint64 __v; \
9      asm volatile("csrr __, " #csr \
10                  : "=r"(__v): \
11                  : "memory") \
12      __v; \
13  })
14
15 #define csr_write(csr, val) \
16 ({ \
17     uint64 __v = (uint64)(val); \
18     asm volatile ("csrw " #csr " , %0" \
19                 : : "r" (__v) \
20                 : "memory"); \
21 })
22
23 #endif
24

```

2.5 开启Trap处理

1. 设置 `stvec`，将 `_traps` (`_trap` 在 4.3 中实现) 所表示的地址写入 `stvec`，这里我们采用 `Direct` 模式，而 `_traps` 则是 trap 处理入口函数的基地址。
2. 开启时钟中断，将 `sie[STIE]` 置 1。
3. 设置第一次时钟中断，参考 `clock_set_next_event()` (`clock_set_next_event()` 在 4.3.4 中介绍) 中的逻辑用汇编实现。
4. 开启 S 态下的中断响应，将 `sstatus[SIE]` 置 1。

实现如下：主要利用 `csrr` 和 `csrw` 函数读写控制寄存器，首先制作立即数，再用立即数去和相应的控制寄存器进行操作来进行置位。

Head.S

```

1  .extern start_kernel
2
3  .section .text.init
4  .globl _start
5  _start:
6      la sp, boot_stack_top
7      #---
8      # set stvec
9      la t2, _traps
10     csrw stvec, t2
11     #---
12     #set sie
13     li t5, 0x20
14     csrr t3, sie
15     or t3, t3, t5
16     csrw sie, t3
17     #---
18     # set interrupt
19     call clock_set_next_event
20     #----
21     # set sstatus
22     li t6, 0x2
23     csrr t3, sstatus
24     or t3, t3, t6
25     csrw sstatus, t3
26     #---
27     jal start_kernel
28     #---
29     .section .bss.stack
30     .globl boot_stack
31 boot_stack:
32     .space 4096 # <-- change to your stack size
33
34     .globl boot_stack_top
35 boot_stack_top:
36

```

2.6 实现上下文切换

在这一部分我们需要做如下的这样一些操作

1. save 32 registers and sepc to stack
2. call trap_handler
3. restore sepc and 32 registers (x2(sp) should be restore last) from stack
4. return from trap

其中：在栈上的相关操作采用sd/ld指令，读写控制状态寄存器采用csrr/csrw，传递参数用a0, a1寄存器传递参数

Entry.S

```

1  .section .text.entry
2  .align 2
3  .globl _traps

```

```

4  _traps:
5      # YOUR CODE HERE
6      # -----
7
8      # 1. save 32 registers and sepc to stack
9      sd x0, -8(x2)
10     sd x1, -16(x2)
11     sd x2, -24(x2)
12     sd x3, -32(x2)
13     sd x4, -40(x2)
14     sd x5, -48(x2)
15     sd x6, -56(x2)
16     sd x7, -64(x2)
17     sd x8, -72(x2)
18     sd x9, -80(x2)
19     sd x10, -88(x2)
20     sd x11, -96(x2)
21     sd x12, -104(x2)
22     sd x13, -112(x2)
23     sd x14, -120(x2)
24     sd x15, -128(x2)
25     sd x16, -136(x2)
26     sd x17, -144(x2)
27     sd x18, -152(x2)
28     sd x19, -160(x2)
29     sd x20, -168(x2)
30     sd x21, -176(x2)
31     sd x22, -184(x2)
32     sd x23, -192(x2)
33     sd x24, -200(x2)
34     sd x25, -208(x2)
35     sd x26, -216(x2)
36     sd x27, -224(x2)
37     sd x28, -232(x2)
38     sd x29, -240(x2)
39     sd x30, -248(x2)
40     sd x31, -256(x2)
41
42     csrr t0, sepc
43     sd t0, -264(x2)
44     addi x2, x2, -264
45     # -----
46
47     # 2. call trap_handler
48     csrr a0, scause
49     csrr a1, sepc
50     call trap_handler
51     # -----
52
53     # 3. restore sepc and 32 registers (x2(sp) should be restore last)
    from stack
54     ld t0, 0(x2)
55     csrw sepc, t0
56
57     ld x31, 8(x2)

```

```

58         ld x30, 16(x2)
59         ld x29, 24(x2)
60         ld x28, 32(x2)
61         ld x27, 40(x2)
62         ld x26, 48(x2)
63         ld x25, 56(x2)
64         ld x24, 64(x2)
65         ld x23, 72(x2)
66         ld x22, 80(x2)
67         ld x21, 88(x2)
68         ld x20, 96(x2)
69         ld x19, 104(x2)
70         ld x18, 112(x2)
71         ld x17, 120(x2)
72         ld x16, 128(x2)
73         ld x15, 136(x2)
74         ld x14, 144(x2)
75         ld x13, 152(x2)
76         ld x12, 160(x2)
77         ld x11, 168(x2)
78         ld x10, 176(x2)
79         ld x9, 184(x2)
80         ld x8, 192(x2)
81         ld x7, 200(x2)
82         ld x6, 208(x2)
83         ld x5, 216(x2)
84         ld x4, 224(x2)
85         ld x3, 232(x2)
86         ld x1, 248(x2)
87         ld x0, 256(x2)
88         ld x2, 240(x2)
89         # -----
90
91         # 4. return from trap
92         sret
93         # -----

```

2.7 实现Trap处理函数

- 在这一部分，我们对传入的scause进行位判断，经过查表，我们发现当scause的最高位是1以及其他位所表示的数为0x5时，表示中断类型是Supervisor timer interrupt，所以我们对此进行判断即可

```

1  // trap.c
2  #include "sbi.h"
3  #include "printk.h"
4
5  void trap_handler(unsigned long scause, unsigned long sepc) {
6      // 通过 `scause` 判断trap类型
7      // 如果是interrupt 判断是否是timer interrupt
8      // 如果是timer interrupt 则打印输出相关信息，并通过 `clock_set_next_event()`
        设置下一次时钟中断
9      // `clock_set_next_event()` 见 4.3.4 节
10     // 其他interrupt / exception 可以直接忽略
11

```

```

12 // YOUR CODE HERE
13 // (scause&1UL << 63)&&((scause&0x5)==0x5)
14 if((scause&(1ULL<<63)) && (scause&0x5 == 0x5))
15 {
16     // printk("Supervisor Mode Timer Interrupt\n");
17     clock_set_next_event();
18 }
19 else
20 {
21     printk("Other Interrupt\n");
22 }
23 }

```

2.8 实现时钟中断相关函数

在这一部分中，我们只用在汇编中用rdtime指令读取寄存器得到ctime寄存器中的值，在clock_set_next_event()函数中通过sbi_ecall接口调用Opensbi的sbi_set_timer即可

```

1 // clock.c
2 #include "types.h"
3 #include "printk.h"
4 // QEMU中时钟的频率是10MHz，也就是1秒钟相当于10000000个时钟周期。
5 unsigned long TIMECLOCK = 10000000;
6
7 unsigned long get_cycles() {
8     // 编写内联汇编，使用 rdtime 获取 time 寄存器中（也就是mtime 寄存器）的值并返回
9     // YOUR CODE HERE
10    unsigned long r_time = 0;
11    __asm__ volatile(
12        "rdtime t0\n"
13        "mv %[ret], t0"
14        : [ret]="r"(r_time)
15        :
16        : "memory"
17    );
18    return r_time;
19 }
20
21 void clock_set_next_event() {
22     // 下一次 时钟中断 的时间点
23     unsigned long next = get_cycles() + TIMECLOCK;
24     printk("kernel is running\n");
25     printk("[S] Supervisor Mode Timer Interrupt\n");
26     // 使用 sbi_ecall 来完成对下一次时钟中断的设置
27     // YOUR CODE HERE
28     sbi_ecall(0x0, 0x0, next, 0, 0, 0, 0, 0);
29 }
30

```

2.9 编译以及测试

经过编译，可以看到每隔1s，虚拟机便会触发中断，输出相应内容

[illegible]

三、讨论和心得

在本次的实验中，我学习了基本的Risc-V汇编语言编写，熟悉了相应的语法。了解了OpenSBI在实验中起到的接口作用，它作为Bootloader完成机器启动时 M-mode 下的硬件初始化与寄存器设置，并提供相应接口以便我们在S-mode下可以操作M-mode相应寄存器的值。除此之外，我也重新系统学习了Makefile的编写，了解了RISC-V的Trap处理如何进行编写，并熟悉了CPU的上下文切换机制。

四、思考题

1. 请总结一下 RISC-V 的 calling convention, 并解释 Caller / Callee Saved Register 有什么区别?

RISC-V的函数调用如下:

- 数据对齐：低精度数据保存至寄存器中进行相应拓展，RV64将扩展至64位
- RISC-V通过call指令来调用编写好的函数
- 尽可能的使用寄存器来传递参数，其中包括a0-a7整数寄存器，其中a0-a1可用来传递函数返回值，以及fa0-fa7浮点数寄存器，其中fa0-fa7可用来传递返回值。2个指针字长的返回值分别放入a0与a1
- 对于不同的参数有着不同的传递方式，整型寄存器通过ai寄存器来传递，浮点寄存器通过fai寄存器传递，结构体的每个字段会按照指针长度对齐，参数寄存器保存结构体头部8个指针字长的数据

- 对于小于一个指针字的参数，通过寄存器的最低有效位传递，或者通过栈传递保存在指针字的低位。对于等于两个指针字的参数，通过栈传递时自然对齐。更长的参数通过reference传递
- 栈传递时向下增长

Caller / Callee Saved Register 区别：

- Caller Saver Register：在callee函数运行时，这些寄存器的值可能被破坏，但无需由callee自身保存，而由caller进行这些寄存器值的保存
- Callee Saved Register：在caller调用callee的时候，这些寄存器的值需要在callee执行前进行保存，并在callee

2. 编译之后，通过 System.map 查看 vmlinux.lds 中自定义符号的值（截图）。

编译完成后，在lab1根目录下查看System.map中自定义符号的值

```
● squhuang@squhuang-virtual-machine:~/os23fall-stu/src/lab1$ cat System.map
0000000080200000 A BASE_ADDR
0000000080203000 B boot_stack
0000000080204000 B boot_stack_top
0000000080200190 T clock_set_next_event
0000000080204000 B _ebss
0000000080202008 D _edata
0000000080204000 B _kernel
00000000802010c8 R _erodata
0000000080200940 T _etext
0000000080200160 T get_cycles
0000000080202008 d _GLOBAL_OFFSET_TABLE_
00000000802008c0 T printk
0000000080200394 T putc
000000008020020c T sbi_ecall
0000000080203000 B _sbss
0000000080202000 D _sdata
0000000080200000 T _skernel
0000000080201000 R _srodata
0000000080200000 T _start
0000000080200340 T start_kernel
0000000080200000 T _stext
0000000080200384 T test
0000000080202000 D TIMECLOCK
00000000802002e8 T trap_handler
000000008020003c T _traps
00000000802003e4 t vprintfmt
```

3. 用 `csr_read` 宏读取 `sstatus` 寄存器的值，对照 RISC-V 手册解释其含义（截图）。

```
int start_kernel() {
    printk("2022");
    printk(" Hello RISC-V\n");
    // sbi_ecall(0x1, 0x0, 0x41, 0, 0, 0, 0, 0);
    char p[65] = "0000000000000000000000000000000000000000000000000000000000000000";
    unsigned long long a = csr_read(sstatus);
    int i = 0;
    while(a != 0)
    {
        p[i++] = a % 2 + '0';
        // putc(p[i-1]);
        a /= 2;
    }
    //From 0 - i-1
    for(int j = 0; j <= (i-1)/2; j++)
    {
        int temp = p[j];
        p[j] = p[i-1-j];
        p[i-1-j] = temp;
    }
    printk(p);
    printk("\n");
    test(); // DO NOT DELETE !!!

    return 0;
}
```

[illegible]

31		30		20				19		18		17			
SD		WPRI				MXR		SUM		WPRI					
1		11				1		1		1					

16		15		14		13		12		11		10		9		8		7		6		5		4		2		1		0	
XS[1:0]		FS[1:0]		WPRI		VS[1:0]		SPP		WPRI		UBE		SPIE		WPRI		SIE		WPRI											
2		2		2		2		1		1		1		1		3		1		1											

63	62							34	33	32	31			20	19	18	17	
SD	WPRI								UXL[1:0]		WPRI				MXR	SUM	WPRI	
1	29								2		12				1	1	1	

	16	15	14	13	12	11	10	9	8	7	6	5	4	2	1	0	
XS[1:0]	FS[1:0]		WPRI		VS[1:0]		SPP		WPRI		UBE	SPIE	WPRI		SIE	WPRI	
2	2		2		2		1		1	1	1	1	3		1	1	

UIE = 0: U-mode下禁止中断

UBE = 0: U-mode下进行的显示内存访问为小端

4. 用 `csr_write` 宏向 `sscratch` 寄存器写入数据，并验证是否写入成功（截图）。

修改main.c如下图所示，向sscratch寄存器中写入0x88880000

```
// bi_ecall(0x1, 0x0, 0x41, 0, 0, 0, 0);  
char p[65] = "0000000000000000000000000000000000000000000000000000000000000000";  
// unsigned long long a = csr_read(sscratch);  
unsigned long long num = 0x88880000;  
csr_write(sscratch, num);  
unsigned long long a = csr_read(sscratch);  
int i = 0;  
while(a != 0)  
{  
    p[i++] = a % 2 + '0';  
    // putc(p[i-1]);  
    a /= 2;  
}  
//From 0 - i-1  
for(int j = 0; j <= (i-1)/2; j++)  
{  
    int temp = p[j];  
    p[j] = p[i-1-j];  
    p[i-1-j] = temp;  
}  
printfk(p);  
printfk("\n");  
test(); // DO NOT DELETE !!!  
  
return 0;
```

可以查看到 `sscratch` 寄存器已经变为 `0x88880000`

[illegible]

5. Detail your steps about how to get `arch/arm64/kernel/sys.i`

- 运行如下指令搜索linux下用于arm64的交叉编译器，并进行安装，选择gcc-10版本，创建软链接到aarch64-linux-gnu-gcc

```
1 | sudo ln -sf aarch64-linux-gnu-gcc-10 aarch64-linux-gnu-gcc
```

```
squhuang@squhuang-virtual-machine:~/qemu-8.0.5/docs$ apt-cache search arm64 | grep gcc-
gcc-9-aarch64-linux-gnu - GNU C compiler (cross compiler for arm64 architecture)
gcc-aarch64-linux-gnu - GNU C compiler for the arm64 architecture
libgcc-9-dev-arm64-cross - GCC support library (development files)
libgcc-s1-arm64-cross - GCC support library (arm64)
gcc-10-aarch64-linux-gnu - GNU C compiler (cross compiler for arm64 architecture)
gcc-8-aarch64-linux-gnu - GNU C compiler (cross compiler for arm64 architecture)
libgcc-10-dev-arm64-cross - GCC support library (development files)
libgcc-8-dev-arm64-cross - GCC support library (development files)
```

- 在之前下载的linux目录下面执行如下命令

```
1 make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- defconfig
2 make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
  /arch/arm64/kernel/sys.i
```

此处显示已经编译好了 `sys.i` 文件

```
HOSTCC  scripts/mod/sumversion.o
HOSTLD  scripts/mod/modpost
CC      kernel/bounds.s
CC      arch/arm64/kernel/asm-offsets.s
CALL    scripts/checksyscalls.sh
CHKSHA1 include/linux/atomic/atomic-arch-fallback.h
CHKSHA1 include/linux/atomic/atomic-instrumented.h
CHKSHA1 include/linux/atomic/atomic-long.h
LDS      arch/arm64/kernel/vdso/vdso.lds
CC      arch/arm64/kernel/vdso/vgettimeofday.o
AS      arch/arm64/kernel/vdso/note.o
AS      arch/arm64/kernel/vdso/sigreturn.o
LD       arch/arm64/kernel/vdso/vdso.so.dbg
VDSOSYM include/generated/vdso-offsets.h
OBJCOPY arch/arm64/kernel/vdso/vdso.so
CPP      arch/arm64/kernel/sys.i
squhuang@squhuang-virtual-machine:~/linux-6.5.4$ ls
```

- 可以搜索到当前目录存在 `sys.i`

```
squhuang@squhuang-virtual-machine:~/linux-6.5.4/arch/arm64/kernel$ ls | grep *.i
sys.i
```

- Find system call table of Linux v6.0 for `ARM32`, `RISC-V(32 bit)`, `RISC-V(64 bit)`, `x86(32 bit)`, `x86_64`

List source code file, the whole system call table with macro expanded, screenshot every step.

- 首先下载linux-6.0.1源码并解压，解压后，在 `~/linux-6.0.1/arch/example-arch` 下进行有关文件的搜索

```

● squhuang@squhuang-virtual-machine:~/linux-6.0.1/arch/arm$ find . -name sys*
./mach-mvebu/system-controller.c
./tools/syscall.tbl
./tools/syscallnr.sh
./kernel/sys_arm.c
./kernel/sys_oabi-compat.c
./mach-highbank/system.c
./mach-highbank/sysregs.h
./mach-imx/system.c
./include/asm/syscall.h
./include/asm/system_misc.h
./include/asm/system_info.h

```

- arm32: 文件是 `~/linux-6.0.1/arch/arm/tools/syscall.tbl`,

完整调用表可以见附件

```

#
0      common  restart_syscall      sys_restart_syscall
1      common  exit                sys_exit
2      common  fork                sys_fork
3      common  read                sys_read
4      common  write               sys_write
5      common  open                sys_open
6      common  close               sys_close
# 7 was sys_waitpid
8      common  creat               sys_creat
9      common  link                sys_link
10     common  unlink              sys_unlink
11     common  execve              sys_execve
12     common  chdir               sys_chdir
13     oabi    time                sys_time32
14     common  mknod               sys_mknod
15     common  chmod               sys_chmod
16     common  lchown              sys_lchown16
# 17 was sys_break
# 18 was sys_stat
19     common  lseek               sys_lseek
20     common  getpid              sys_getpid
21     common  mount               sys_mount
22     oabi    umount              sys_oldumount

```

- RISC-V_32/64: 文件应位于 `~/linux-6.5.4/include/uapi/asm-generic/unisted.h` 中, 与其他体系结构共享一个通用的系统调用表, 如下图所示

```

#define __SC_COMP(_nr, _sys, _comp) __SYSCALL(_nr, _sys)
#define __SC_COMP_3264(_nr, _32, _64, _comp) __SC_3264(_nr, _32, _64)
#endif

#define __NR_io_setup 0
__SC_COMP(__NR_io_setup, sys_io_setup, compat_sys_io_setup)
#define __NR_io_destroy 1
__SYSCALL(__NR_io_destroy, sys_io_destroy)
#define __NR_io_submit 2
__SC_COMP(__NR_io_submit, sys_io_submit, compat_sys_io_submit)
#define __NR_io_cancel 3
__SYSCALL(__NR_io_cancel, sys_io_cancel)
#if defined(__ARCH_WANT_TIME32_SYSCALLS) || __BITS_PER_LONG != 32
#define __NR_io_getevents 4
__SC_3264(__NR_io_getevents, sys_io_getevents_time32, sys_io_getevents)
#endif

/* fs/xattr.c */
#define __NR_setxattr 5
__SYSCALL(__NR_setxattr, sys_setxattr)
#define __NR_lsetxattr 6
__SYSCALL(__NR_lsetxattr, sys_lsetxattr)
#define __NR_fsetxattr 7

```

- x86_32: 文件位于 `./entry/syscalls/syscall_32.tbl`

```

# 32-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point> <compat entry point>
#
# The __ia32_sys and __ia32_compat_sys stubs are created on-the-fly for
# sys_*( ) system calls and compat_sys_*( ) compat system calls if
# IA32_EMULATION is defined, and expect struct pt_regs *regs as their only
# parameter.
#
# The abi is always "i386" for this file.
#
0   i386   restart_syscall   sys_restart_syscall
1   i386   exit              sys_exit
2   i386   fork              sys_fork
3   i386   read              sys_read
4   i386   write             sys_write
5   i386   open              sys_open          compat_sys_open
6   i386   close             sys_close
7   i386   waitpid           sys_waitpid
8   i386   creat             sys_creat
9   i386   link              sys_link

```

- x86_64: 文件位于 `./entry/syscalls/syscall_64.tbl`

```
# 64-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The __x64_sys_*() stubs are created on-the-fly for sys_*() system calls
#
# The abi is "common", "64" or "x32" for this file.
#
0 common read sys_read
1 common write sys_write
2 common open sys_open
3 common close sys_close
4 common stat sys_newstat
5 common fstat sys_newfstat
6 common lstat sys_newlstat
7 common poll sys_poll
8 common lseek sys_lseek
9 common mmap sys_mmap
10 common mprotect sys_mprotect
11 common munmap sys_munmap
12 common brk sys_brk
```

7. Explain what is ELF file? Try readelf and objdump command on an ELF file, give screenshot of the output.

Run an ELF file and cat `/proc/PID/maps` to give its memory layout.

- ELF file: ELF全称是Executable and Linkable Format，即可执行可链接文件，是一种二进制文件格式，用于在类Unix系统中存储可执行程序、共享库以及充当目标文件。vmlinux即为一种ELF file
- 使用 `readelf -s vmlinux` 查看lab1内核编译出的vmlinux的symbol table

```
squhuang@squhuang-virtual-machine:~/os23fall-stu/src/lab1$ readelf -s vmlinux
```

Symbol table '.symtab' contains 48 entries:

Num:	Value	Size	Type	Bind	Vis	Ndx	Name
0:	0000000000000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	0000000008020000	0	SECTION	LOCAL	DEFAULT	1	
2:	0000000008020100	0	SECTION	LOCAL	DEFAULT	2	
3:	0000000008020200	0	SECTION	LOCAL	DEFAULT	3	
4:	0000000008020208	0	SECTION	LOCAL	DEFAULT	4	
5:	0000000008020220	0	SECTION	LOCAL	DEFAULT	5	
6:	0000000008020300	0	SECTION	LOCAL	DEFAULT	6	
7:	0000000000000000	0	SECTION	LOCAL	DEFAULT	7	
8:	0000000000000000	0	SECTION	LOCAL	DEFAULT	8	
9:	0000000000000000	0	SECTION	LOCAL	DEFAULT	9	
10:	0000000000000000	0	SECTION	LOCAL	DEFAULT	10	
11:	0000000000000000	0	SECTION	LOCAL	DEFAULT	11	
12:	0000000000000000	0	SECTION	LOCAL	DEFAULT	12	

- 使用 `objdump -t vmlinux` 查看lab1内核编译出的vmlinux的symbol table

具体来说，在medeleg和mideleg中set bit的时候，会将S-mode或U-mode中的相应Trap委托给S-mode trap handler。

- 具体含义：

MEDELEG: 意味着mcause寄存器中的S/M-Mode software interrupt, U Mode external interrupt都为1，代表着这些interrupt都可以被委派到更低权限的trap handler进行处理

MIDELEG: 意味着mip寄存器中的SSIP, STIP, SEIP都为1，即S-mode software interrupts, S-mode timer interrupts, S-mode external interrupts的trap delegation都被开启了，即当前的interrupt都可以在S-mode下进行处理

五、附录

1. arm系统调用表：

```
1  #
2  # Linux system call numbers and entry vectors
3  #
4  # The format is:
5  # <num> <abi>  <name>          [<entry point>          [<oabi compat entry
   point>]]
6  #
7  # where abi is:
8  #  common - for system calls shared between oabi and eabi (may have compat)
9  #  oabi   - for oabi-only system calls (may have compat)
10 #  eabi   - for eabi-only system calls
11 #
12 # For each syscall number, "common" is mutually exclusive with oabi and
   eabi
13 #
14 0  common restart_syscall    sys_restart_syscall
15 1  common exit              sys_exit
16 2  common fork              sys_fork
17 3  common read              sys_read
18 4  common write             sys_write
19 5  common open              sys_open
20 6  common close             sys_close
21 # 7 was sys_waitpid
22 8  common creat              sys_creat
23 9  common link              sys_link
24 10 common unlink            sys_unlink
25 11 common execve            sys_execve
26 12 common chdir             sys_chdir
27 13 oabi  time                sys_time32
28 14 common mknod              sys_mknod
29 15 common chmod             sys_chmod
30 16 common lchown            sys_lchown16
31 # 17 was sys_break
32 # 18 was sys_stat
33 19 common lseek              sys_lseek
34 20 common getpid            sys_getpid
35 21 common mount             sys_mount
36 22 oabi  umount             sys_oldumount
37 23 common setuid            sys_setuid16
```

38	24	common	getuid	sys_getuid16
39	25	oabi	stime	sys_stime32
40	26	common	ptrace	sys_ptrace
41	27	oabi	alarm	sys_alarm
42	# 28 was sys_fstat			
43	29	common	pause	sys_pause
44	30	oabi	utime	sys_utime32
45	# 31 was sys_stty			
46	# 32 was sys_gtty			
47	33	common	access	sys_access
48	34	common	nice	sys_nice
49	# 35 was sys_ftime			
50	36	common	sync	sys_sync
51	37	common	kill	sys_kill
52	38	common	rename	sys_rename
53	39	common	mkdir	sys_mkdir
54	40	common	rmdir	sys_rmdir
55	41	common	dup	sys_dup
56	42	common	pipe	sys_pipe
57	43	common	times	sys_times
58	# 44 was sys_prof			
59	45	common	brk	sys_brk
60	46	common	setgid	sys_setgid16
61	47	common	getgid	sys_getgid16
62	# 48 was sys_signal			
63	49	common	geteuid	sys_geteuid16
64	50	common	getegid	sys_getegid16
65	51	common	acct	sys_acct
66	52	common	umount2	sys_umount
67	# 53 was sys_lock			
68	54	common	ioctl	sys_ioctl
69	55	common	fcntl	sys_fcntl
70	# 56 was sys_mpx			
71	57	common	setpgid	sys_setpgid
72	# 58 was sys_ulimit			
73	# 59 was sys_olduname			
74	60	common	umask	sys_umask
75	61	common	chroot	sys_chroot
76	62	common	ustat	sys_ustat
77	63	common	dup2	sys_dup2
78	64	common	getppid	sys_getppid
79	65	common	getpgrp	sys_getpgrp
80	66	common	setsid	sys_setsid
81	67	common	sigaction	sys_sigaction
82	# 68 was sys_sgetmask			
83	# 69 was sys_ssetmask			
84	70	common	setreuid	sys_setreuid16
85	71	common	setregid	sys_setregid16
86	72	common	sigsuspend	sys_sigsuspend
87	73	common	sigpending	sys_sigpending
88	74	common	sethostname	sys_sethostname
89	75	common	setrlimit	sys_setrlimit
90	# Back compat 2GB limited rlimit			
91	76	oabi	getrlimit	sys_old_getrlimit
92	77	common	getrusage	sys_getrusage

93	78	common	gettimeofday	sys_gettimeofday	
94	79	common	settimeofday	sys_settimeofday	
95	80	common	getgroups	sys_getgroups16	
96	81	common	setgroups	sys_setgroups16	
97	82	oabi	select	sys_old_select	
98	83	common	symlink	sys_symlink	
99	# 84 was sys_lstat				
100	85	common	readlink	sys_readlink	
101	86	common	uselib	sys_uselib	
102	87	common	swapon	sys_swapon	
103	88	common	reboot	sys_reboot	
104	89	oabi	readdir	sys_old_readdir	
105	90	oabi	mmap	sys_old_mmap	
106	91	common	munmap	sys_munmap	
107	92	common	truncate	sys_truncate	
108	93	common	ftruncate	sys_ftruncate	
109	94	common	fchmod	sys_fchmod	
110	95	common	fchown	sys_fchown16	
111	96	common	getpriority	sys_getpriority	
112	97	common	setpriority	sys_setpriority	
113	# 98 was sys_profiling				
114	99	common	statfs	sys_statfs	
115	100	common	fstatfs	sys_fstatfs	
116	# 101 was sys_ioperm				
117	102	oabi	socketcall	sys_socketcall	sys_oabi_socketcall
118	103	common	syslog	sys_syslog	
119	104	common	setitimer	sys_setitimer	
120	105	common	getitimer	sys_getitimer	
121	106	common	stat	sys_newstat	
122	107	common	lstat	sys_newlstat	
123	108	common	fstat	sys_newfstat	
124	# 109 was sys_uname				
125	# 110 was sys_iopl				
126	111	common	vhangup	sys_vhangup	
127	# 112 was sys_idle				
128	# syscall to call a syscall!				
129	113	oabi	syscall	sys_syscall	
130	114	common	wait4	sys_wait4	
131	115	common	swapoff	sys_swapoff	
132	116	common	sysinfo	sys_sysinfo	
133	117	oabi	ipc	sys_ipc	sys_oabi_ipc
134	118	common	fsync	sys_fsync	
135	119	common	sigreturn	sys_sigreturn_wrapper	
136	120	common	clone	sys_clone	
137	121	common	setdomainname	sys_setdomainname	
138	122	common	uname	sys_newuname	
139	# 123 was sys_modify_ldt				
140	124	common	adjtimex	sys_adjtimex_time32	
141	125	common	mprotect	sys_mprotect	
142	126	common	sigprocmask	sys_sigprocmask	
143	# 127 was sys_create_module				
144	128	common	init_module	sys_init_module	
145	129	common	delete_module	sys_delete_module	
146	# 130 was sys_get_kernel_syms				
147	131	common	quotactl	sys_quotactl	

148	132	common	getpgid	sys_getpgid
149	133	common	fchdir	sys_fchdir
150	134	common	bdflush	sys_ni_syscall
151	135	common	sysfs	sys_sysfs
152	136	common	personality	sys_personality
153	# 137 was sys_afs_syscall			
154	138	common	setfsuid	sys_setfsuid16
155	139	common	setfsgid	sys_setfsgid16
156	140	common	_llseek	sys_llseek
157	141	common	getdents	sys_getdents
158	142	common	_newselect	sys_select
159	143	common	flock	sys_flock
160	144	common	msync	sys_msync
161	145	common	readv	sys_readv
162	146	common	writew	sys_writew
163	147	common	getsid	sys_getsid
164	148	common	fdatasync	sys_fdatasync
165	149	common	_sysctl	sys_ni_syscall
166	150	common	mlock	sys_mlock
167	151	common	munlock	sys_munlock
168	152	common	mlockall	sys_mlockall
169	153	common	munlockall	sys_munlockall
170	154	common	sched_setparam	sys_sched_setparam
171	155	common	sched_getparam	sys_sched_getparam
172	156	common	sched_setscheduler	sys_sched_setscheduler
173	157	common	sched_getscheduler	sys_sched_getscheduler
174	158	common	sched_yield	sys_sched_yield
175	159	common	sched_get_priority_max	sys_sched_get_priority_max
176	160	common	sched_get_priority_min	sys_sched_get_priority_min
177	161	common	sched_rr_get_interval	sys_sched_rr_get_interval_time32
178	162	common	nanosleep	sys_nanosleep_time32
179	163	common	mremap	sys_mremap
180	164	common	setresuid	sys_setresuid16
181	165	common	getresuid	sys_getresuid16
182	# 166 was sys_vm86			
183	# 167 was sys_query_module			
184	168	common	poll	sys_poll
185	169	common	nfsservctl	
186	170	common	setresgid	sys_setresgid16
187	171	common	getresgid	sys_getresgid16
188	172	common	prctl	sys_prctl
189	173	common	rt_sigreturn	sys_rt_sigreturn_wrapper
190	174	common	rt_sigaction	sys_rt_sigaction
191	175	common	rt_sigprocmask	sys_rt_sigprocmask
192	176	common	rt_sigpending	sys_rt_sigpending
193	177	common	rt_sigtimedwait	sys_rt_sigtimedwait_time32
194	178	common	rt_sigqueueinfo	sys_rt_sigqueueinfo
195	179	common	rt_sigsuspend	sys_rt_sigsuspend
196	180	common	pread64	sys_pread64 sys_oabi_pread64
197	181	common	pwrite64	sys_pwrite64 sys_oabi_pwrite64
198	182	common	chown	sys_chown16
199	183	common	getcwd	sys_getcwd
200	184	common	capget	sys_capget
201	185	common	capset	sys_capset
202	186	common	sigaltstack	sys_sigaltstack

203	187 common	sendfile	sys_sendfile	
204	# 188 reserved			
205	# 189 reserved			
206	190 common	vfork	sys_vfork	
207	# sus compliant	getrlimit		
208	191 common	ugetrlimit	sys_getrlimit	
209	192 common	mmap2	sys_mmap2	
210	193 common	truncate64	sys_truncate64	sys_oabi_truncate64
211	194 common	ftruncate64	sys_ftruncate64	sys_oabi_ftruncate64
212	195 common	stat64	sys_stat64	sys_oabi_stat64
213	196 common	lstat64	sys_lstat64	sys_oabi_lstat64
214	197 common	fstat64	sys_fstat64	sys_oabi_fstat64
215	198 common	lchown32	sys_lchown	
216	199 common	getuid32	sys_getuid	
217	200 common	getgid32	sys_getgid	
218	201 common	geteuid32	sys_geteuid	
219	202 common	getegid32	sys_getegid	
220	203 common	setreuid32	sys_setreuid	
221	204 common	setregid32	sys_setregid	
222	205 common	getgroups32	sys_getgroups	
223	206 common	setgroups32	sys_setgroups	
224	207 common	fchown32	sys_fchown	
225	208 common	setresuid32	sys_setresuid	
226	209 common	getresuid32	sys_getresuid	
227	210 common	setresgid32	sys_setresgid	
228	211 common	getresgid32	sys_getresgid	
229	212 common	chown32	sys_chown	
230	213 common	setuid32	sys_setuid	
231	214 common	setgid32	sys_setgid	
232	215 common	setfsuid32	sys_setfsuid	
233	216 common	setfsgid32	sys_setfsgid	
234	217 common	getdents64	sys_getdents64	
235	218 common	pivot_root	sys_pivot_root	
236	219 common	mincore	sys_mincore	
237	220 common	madvise	sys_madvise	
238	221 common	fcntl64	sys_fcntl64	sys_oabi_fcntl64
239	# 222 for tux			
240	# 223 is unused			
241	224 common	gettid	sys_gettid	
242	225 common	readahead	sys_readahead	sys_oabi_readahead
243	226 common	setxattr	sys_setxattr	
244	227 common	lsetxattr	sys_lsetxattr	
245	228 common	fsetxattr	sys_fsetxattr	
246	229 common	getxattr	sys_getxattr	
247	230 common	lgetxattr	sys_lgetxattr	
248	231 common	fgetxattr	sys_fgetxattr	
249	232 common	listxattr	sys_listxattr	
250	233 common	llistxattr	sys_llistxattr	
251	234 common	flistxattr	sys_flistxattr	
252	235 common	removexattr	sys_removexattr	
253	236 common	lremovexattr	sys_lremovexattr	
254	237 common	fremovexattr	sys_fremovexattr	
255	238 common	tkill	sys_tkill	
256	239 common	sendfile64	sys_sendfile64	
257	240 common	futex	sys_futex_time32	

258	241	common	sched_setaffinity	sys_sched_setaffinity	
259	242	common	sched_getaffinity	sys_sched_getaffinity	
260	243	common	io_setup	sys_io_setup	
261	244	common	io_destroy	sys_io_destroy	
262	245	common	io_getevents	sys_io_getevents_time32	
263	246	common	io_submit	sys_io_submit	
264	247	common	io_cancel	sys_io_cancel	
265	248	common	exit_group	sys_exit_group	
266	249	common	lookup_dcookie	sys_lookup_dcookie	
267	250	common	epoll_create	sys_epoll_create	
268	251	common	epoll_ctl	sys_epoll_ctl	sys_oabi_epoll_ctl
269	252	common	epoll_wait	sys_epoll_wait	
270	253	common	remap_file_pages	sys_remap_file_pages	
271	# 254 for set_thread_area				
272	# 255 for get_thread_area				
273	256	common	set_tid_address	sys_set_tid_address	
274	257	common	timer_create	sys_timer_create	
275	258	common	timer_settime	sys_timer_settime32	
276	259	common	timer_gettime	sys_timer_gettime32	
277	260	common	timer_getoverrun	sys_timer_getoverrun	
278	261	common	timer_delete	sys_timer_delete	
279	262	common	clock_settime	sys_clock_settime32	
280	263	common	clock_gettime	sys_clock_gettime32	
281	264	common	clock_getres	sys_clock_getres_time32	
282	265	common	clock_nanosleep	sys_clock_nanosleep_time32	
283	266	common	statfs64	sys_statfs64_wrapper	
284	267	common	fstatfs64	sys_fstatfs64_wrapper	
285	268	common	tgkill	sys_tgkill	
286	269	common	utimes	sys_utimes_time32	
287	270	common	arm_fadvise64_64	sys_arm_fadvise64_64	
288	271	common	pciconfig_iobase	sys_pciconfig_iobase	
289	272	common	pciconfig_read	sys_pciconfig_read	
290	273	common	pciconfig_write	sys_pciconfig_write	
291	274	common	mq_open	sys_mq_open	
292	275	common	mq_unlink	sys_mq_unlink	
293	276	common	mq_timedsend	sys_mq_timedsend_time32	
294	277	common	mq_timedreceive	sys_mq_timedreceive_time32	
295	278	common	mq_notify	sys_mq_notify	
296	279	common	mq_getsetattr	sys_mq_getsetattr	
297	280	common	waitid	sys_waitid	
298	281	common	socket	sys_socket	
299	282	common	bind	sys_bind	sys_oabi_bind
300	283	common	connect	sys_connect	sys_oabi_connect
301	284	common	listen	sys_listen	
302	285	common	accept	sys_accept	
303	286	common	getsockname	sys_getsockname	
304	287	common	getpeername	sys_getpeername	
305	288	common	socketpair	sys_socketpair	
306	289	common	send	sys_send	
307	290	common	sendto	sys_sendto	sys_oabi_sendto
308	291	common	recv	sys_recv	
309	292	common	recvfrom	sys_recvfrom	
310	293	common	shutdown	sys_shutdown	
311	294	common	setsockopt	sys_setsockopt	
312	295	common	getsockopt	sys_getsockopt	

313	296	common	sendmsg	sys_sendmsg	sys_oabi_sendmsg
314	297	common	recvmsg	sys_recvmsg	
315	298	common	semop	sys_semop	sys_oabi_semop
316	299	common	semget	sys_semget	
317	300	common	semctl	sys_old_semctl	
318	301	common	msgsnd	sys_msgsnd	
319	302	common	msgrcv	sys_msgrcv	
320	303	common	msgget	sys_msgget	
321	304	common	msgctl	sys_old_msgctl	
322	305	common	shmat	sys_shmat	
323	306	common	shmdt	sys_shmdt	
324	307	common	shmget	sys_shmget	
325	308	common	shmctl	sys_old_shmctl	
326	309	common	add_key	sys_add_key	
327	310	common	request_key	sys_request_key	
328	311	common	keyctl	sys_keyctl	
329	312	common	semtimedop	sys_semtimedop_time32	sys_oabi_semtimedop
330	313	common	vserver		
331	314	common	ioprio_set	sys_ioprio_set	
332	315	common	ioprio_get	sys_ioprio_get	
333	316	common	inotify_init	sys_inotify_init	
334	317	common	inotify_add_watch	sys_inotify_add_watch	
335	318	common	inotify_rm_watch	sys_inotify_rm_watch	
336	319	common	mbind	sys_mbind	
337	320	common	get_mempolicy	sys_get_mempolicy	
338	321	common	set_mempolicy	sys_set_mempolicy	
339	322	common	openat	sys_openat	
340	323	common	mkdirat	sys_mkdirat	
341	324	common	mknodat	sys_mknodat	
342	325	common	fchownat	sys_fchownat	
343	326	common	futimesat	sys_futimesat_time32	
344	327	common	fstatat64	sys_fstatat64	sys_oabi_fstatat64
345	328	common	unlinkat	sys_unlinkat	
346	329	common	renameat	sys_renameat	
347	330	common	linkat	sys_linkat	
348	331	common	symlinkat	sys_symlinkat	
349	332	common	readlinkat	sys_readlinkat	
350	333	common	fchmodat	sys_fchmodat	
351	334	common	faccessat	sys_faccessat	
352	335	common	pselect6	sys_pselect6_time32	
353	336	common	ppoll	sys_ppoll_time32	
354	337	common	unshare	sys_unshare	
355	338	common	set_robust_list	sys_set_robust_list	
356	339	common	get_robust_list	sys_get_robust_list	
357	340	common	splice	sys_splice	
358	341	common	arm_sync_file_range	sys_sync_file_range2	
359	342	common	tee	sys_tee	
360	343	common	vmsplice	sys_vmsplice	
361	344	common	move_pages	sys_move_pages	
362	345	common	getcpu	sys_getcpu	
363	346	common	epoll_pwait	sys_epoll_pwait	
364	347	common	kexec_load	sys_kexec_load	
365	348	common	utimensat	sys_utimensat_time32	
366	349	common	signalfd	sys_signalfd	
367	350	common	timerfd_create	sys_timerfd_create	

368	351	common	eventfd	sys_eventfd
369	352	common	fallocate	sys_fallocate
370	353	common	timerfd_settime	sys_timerfd_settime32
371	354	common	timerfd_gettime	sys_timerfd_gettime32
372	355	common	signalfd4	sys_signalfd4
373	356	common	eventfd2	sys_eventfd2
374	357	common	epoll_create1	sys_epoll_create1
375	358	common	dup3	sys_dup3
376	359	common	pipe2	sys_pipe2
377	360	common	inotify_init1	sys_inotify_init1
378	361	common	preadv	sys_preadv
379	362	common	pwritev	sys_pwritev
380	363	common	rt_tgsigqueueinfo	sys_rt_tgsigqueueinfo
381	364	common	perf_event_open	sys_perf_event_open
382	365	common	recvmmsg	sys_recvmmsg_time32
383	366	common	accept4	sys_accept4
384	367	common	fanotify_init	sys_fanotify_init
385	368	common	fanotify_mark	sys_fanotify_mark
386	369	common	prlimit64	sys_prlimit64
387	370	common	name_to_handle_at	sys_name_to_handle_at
388	371	common	open_by_handle_at	sys_open_by_handle_at
389	372	common	clock_adjtime	sys_clock_adjtime32
390	373	common	syncfs	sys_syncfs
391	374	common	sendmmsg	sys_sendmmsg
392	375	common	setns	sys_setns
393	376	common	process_vm_readv	sys_process_vm_readv
394	377	common	process_vm_writev	sys_process_vm_writev
395	378	common	kcmp	sys_kcmp
396	379	common	finit_module	sys_finit_module
397	380	common	sched_setattr	sys_sched_setattr
398	381	common	sched_getattr	sys_sched_getattr
399	382	common	renameat2	sys_renameat2
400	383	common	seccomp	sys_seccomp
401	384	common	getrandom	sys_getrandom
402	385	common	memfd_create	sys_memfd_create
403	386	common	bpf	sys_bpf
404	387	common	execveat	sys_execveat
405	388	common	userfaultfd	sys_userfaultfd
406	389	common	membarrier	sys_membarrier
407	390	common	mlock2	sys_mlock2
408	391	common	copy_file_range	sys_copy_file_range
409	392	common	preadv2	sys_preadv2
410	393	common	pwritev2	sys_pwritev2
411	394	common	pkey_mprotect	sys_pkey_mprotect
412	395	common	pkey_alloc	sys_pkey_alloc
413	396	common	pkey_free	sys_pkey_free
414	397	common	statx	sys_statx
415	398	common	rseq	sys_rseq
416	399	common	io_pgetevents	sys_io_pgetevents_time32
417	400	common	migrate_pages	sys_migrate_pages
418	401	common	kexec_file_load	sys_kexec_file_load
419	# 402 is unused			
420	403	common	clock_gettime64	sys_clock_gettime
421	404	common	clock_settime64	sys_clock_settime
422	405	common	clock_adjtime64	sys_clock_adjtime

423	406	common	clock_getres_time64	sys_clock_getres
424	407	common	clock_nanosleep_time64	sys_clock_nanosleep
425	408	common	timer_gettime64	sys_timer_gettime
426	409	common	timer_settime64	sys_timer_settime
427	410	common	timerfd_gettime64	sys_timerfd_gettime
428	411	common	timerfd_settime64	sys_timerfd_settime
429	412	common	utimensat_time64	sys_utimensat
430	413	common	pselect6_time64	sys_pselect6
431	414	common	ppoll_time64	sys_ppoll
432	416	common	io_pgetevents_time64	sys_io_pgetevents
433	417	common	recvmmsg_time64	sys_recvmmsg
434	418	common	mq_timedsend_time64	sys_mq_timedsend
435	419	common	mq_timedreceive_time64	sys_mq_timedreceive
436	420	common	semtimedop_time64	sys_semtimedop
437	421	common	rt_sigtimedwait_time64	sys_rt_sigtimedwait
438	422	common	futex_time64	sys_futex
439	423	common	sched_rr_get_interval_time64	sys_sched_rr_get_interval
440	424	common	pidfd_send_signal	sys_pidfd_send_signal
441	425	common	io_uring_setup	sys_io_uring_setup
442	426	common	io_uring_enter	sys_io_uring_enter
443	427	common	io_uring_register	sys_io_uring_register
444	428	common	open_tree	sys_open_tree
445	429	common	move_mount	sys_move_mount
446	430	common	fsopen	sys_fsopen
447	431	common	fsconfig	sys_fsconfig
448	432	common	fsmount	sys_fsmount
449	433	common	fspick	sys_fspick
450	434	common	pidfd_open	sys_pidfd_open
451	435	common	clone3	sys_clone3
452	436	common	close_range	sys_close_range
453	437	common	openat2	sys_openat2
454	438	common	pidfd_getfd	sys_pidfd_getfd
455	439	common	faccessat2	sys_faccessat2
456	440	common	process_madvise	sys_process_madvise
457	441	common	epoll_pwait2	sys_epoll_pwait2
458	442	common	mount_setattr	sys_mount_setattr
459	443	common	quotactl_fd	sys_quotactl_fd
460	444	common	landlock_create_ruleset	sys_landlock_create_ruleset
461	445	common	landlock_add_rule	sys_landlock_add_rule
462	446	common	landlock_restrict_self	sys_landlock_restrict_self
463	# 447	reserved for memfd_secret		
464	448	common	process_mrelease	sys_process_mrelease
465	449	common	futex_waitv	sys_futex_waitv
466	450	common	set_mempolicy_home_node	sys_set_mempolicy_home_node
467				

