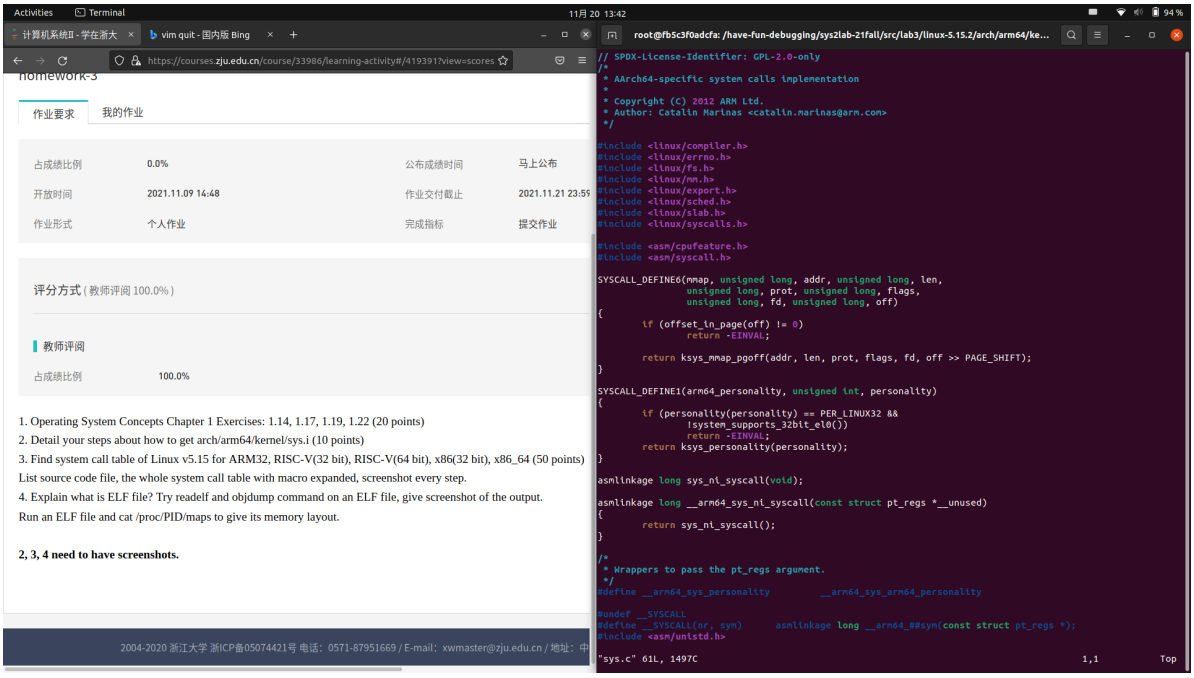
HW3

Q1

1. Interrupts are changes in the flow of the system generated by the hardware. The interrupt handler is called to handle the cause of the interrupt; then, control will return to the context and instructions of the interrupt. A trap is an interrupt generated by software. Interrupts can be used to signal I/O completion, thereby avoiding device polling. Traps can be used to call operating system routines or catch arithmetic errors
2. The operating system of such machines needs to be in control (or monitoring mode) at all times. This can be done in two ways: a. Software interpretation of all user programs (e.g. some BASIC, Java, and LISP systems). The software interpreter will provide content that is not provided by the hardware in the software. b. All programs are required to be written in a high-level language so that all target code is compiler generated. The compiler generates (inline or through function calls) a protection check for hardware loss.
3. f c a d e g b
4. The processor can track the location associated with each process and restrict access to locations outside the scope of the program. Information about the program memory range can be maintained by using basic registers and limit registers and performing checks on each memory access.

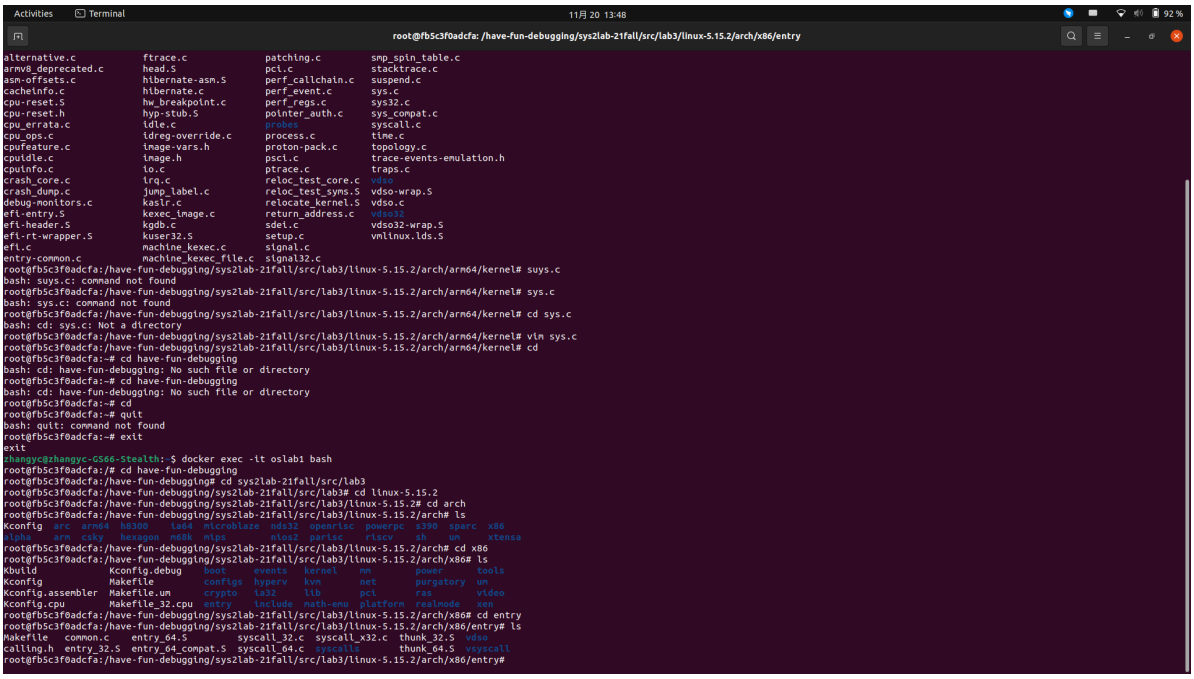
Q2

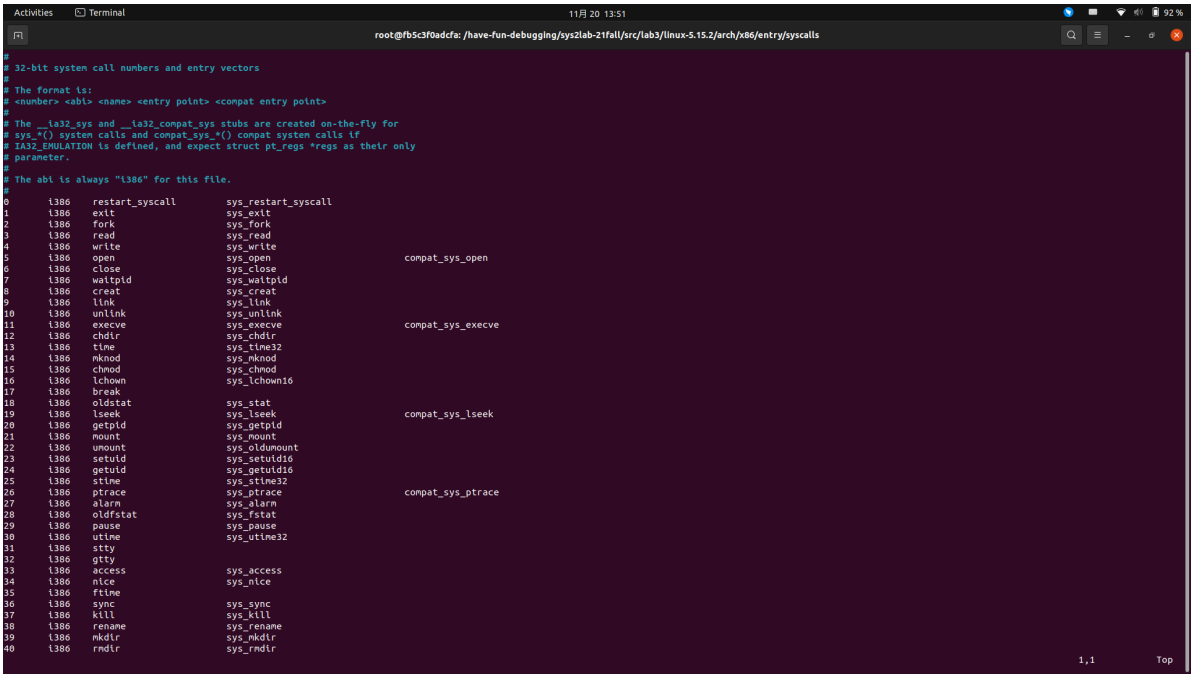
we can use the command ”cd arch/arm64/kernel ”and “vim sys.c” get the code

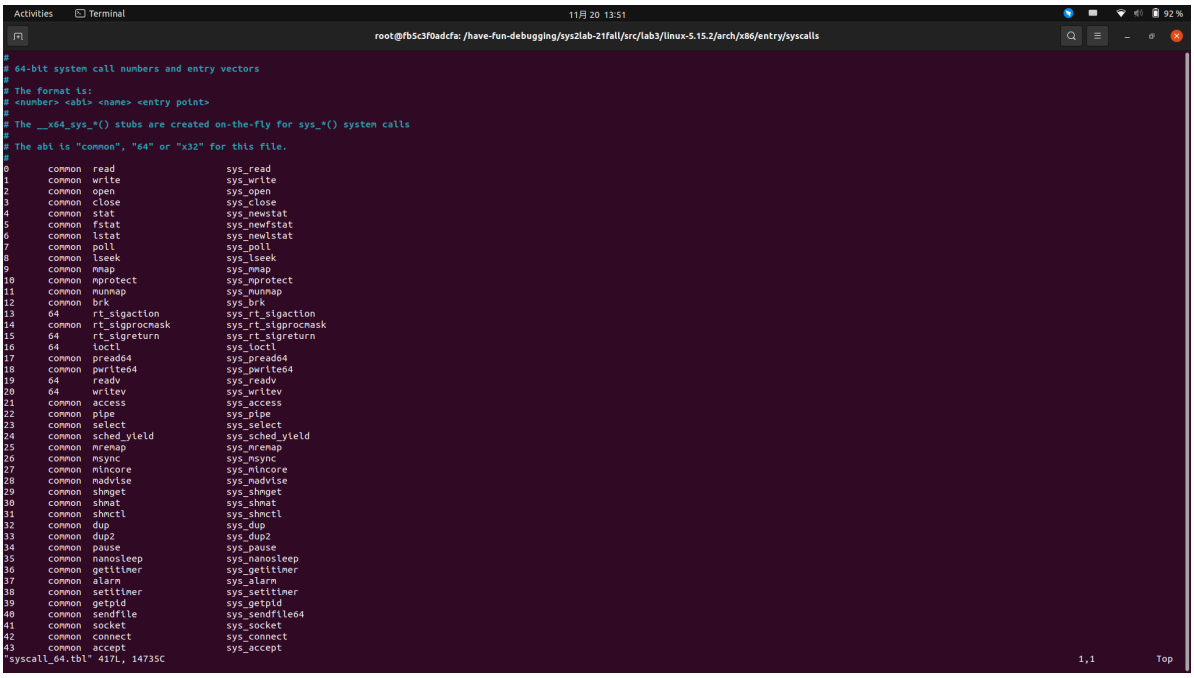


Q3

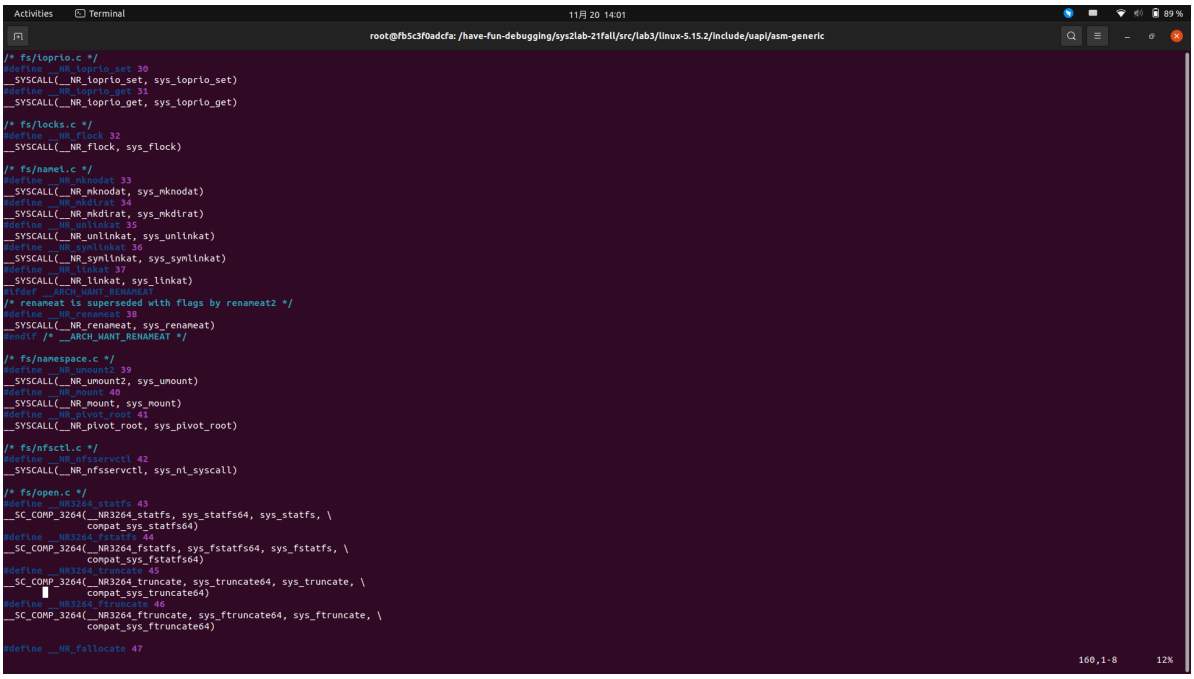
we can get the syscall code on LInux for x86 & x86_64 form arch/x86/entry

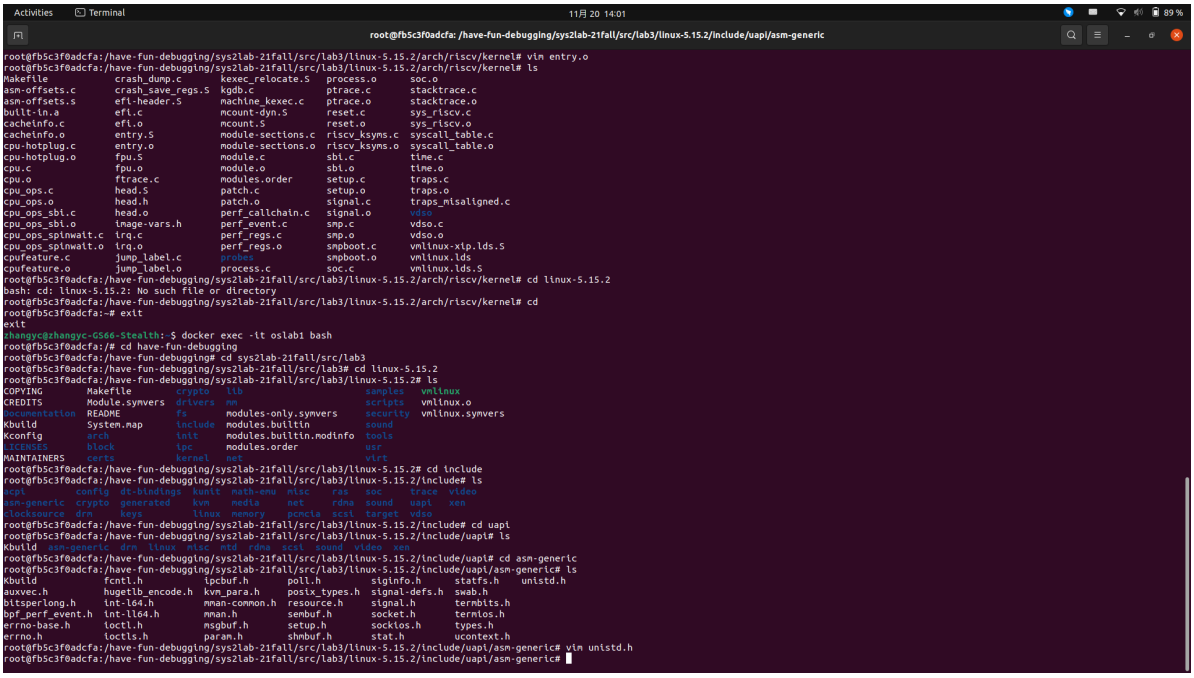

Activities
Terminal
11月 20 13:48
92 %
root@fb5c3f0adcfa: /have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86/entry
alternative.c ftrace.c patching.c smp_spin_table.c
arnv8_deprecated.c head.S pci.c stacktrace.c
asm-offsets.c hibernate-asm.S perf_callchain.c suspend.c
cacheinfo.c hibernate.c perf_event.c sys.c
cpu-reset.S hw_breakpoint.c perf_regs.c sys32.c
cpu-reset.h hyp-stub.S pointer_auth.c sys_compat.c
cpu_errata.c idle.c probes syscall.c
cpu_ops.c idreg-override.c process.c time.c
cpufeature.c image-vars.h proton-pack.c topology.c
cpuidle.c image.h psci.c trace-events-emulation.h
cpuinfo.c io.c ptrace.c traps.c
crash_core.c irq.c reloc_test_core.c vdso
crash_dump.c jump_label.c reloc_test_syms.S vdso-wrap.S
debug-monitors.c kaslr.c relocate_kernel.S vdso.c
efi-entry.S kexec_image.c return_address.c vdso32
efi-header.S kgdb.c sdei.c vdso32-wrap.S
efi-rt-wrapper.S kuser32.S setup.c vmlinux.lds.S
efi.c machine_kexec.c signal.c
entry-common.c machine_kexec_file.c signal32.c
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/arm64/kernel# suys.c
bash: suys.c: command not found
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/arm64/kernel# sys.c
bash: sys.c: command not found
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/arm64/kernel# cd sys.c
bash: cd: sys.c: Not a directory
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/arm64/kernel# vim sys.c
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/arm64/kernel# cd
root@fb5c3f0adcfa:~# cd have-fun-debugging
bash: cd: have-fun-debugging: No such file or directory
root@fb5c3f0adcfa:~# cd have-fun-debugging
bash: cd: have-fun-debugging: No such file or directory
root@fb5c3f0adcfa:~# cd
root@fb5c3f0adcfa:~# quit
bash: quit: command not found
root@fb5c3f0adcfa:~# exit
exit
zhangyc@zhangyc-GS66-Stealth:~$ docker exec -it oslab1 bash
root@fb5c3f0adcfa:/# cd have-fun-debugging
root@fb5c3f0adcfa:/have-fun-debugging# cd sys2lab-21fall/src/lab3
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3# cd linux-5.15.2
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2# cd arch
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch# ls
Kconfig arc arm64 h8300 ia64 microblaze nds32 openrisc powerpc s390 sparc x86
alpha arm csky hexagon m68k mips nios2 parisc riscv sh um xtensa
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch# cd x86
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86# ls
Kbuild Kconfig.debug boot events kernel mm power tools
Kconfig Makefile configs hyperv kvn net purgatory um
Kconfig.assembler Makefile.um crypto ia32 lib pci ras video
Kconfig.cpu Makefile_32.cpu entry include math-emu platform realmode xen
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86# cd entry
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86/entry# ls
Makefile common.c entry_64.S syscall_32.c syscall_x32.c thunk_32.S vdso
calling.h entry_32.S entry_64_compat.S syscall_64.c syscalls thunk_64.S vsyscall
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86/entry#

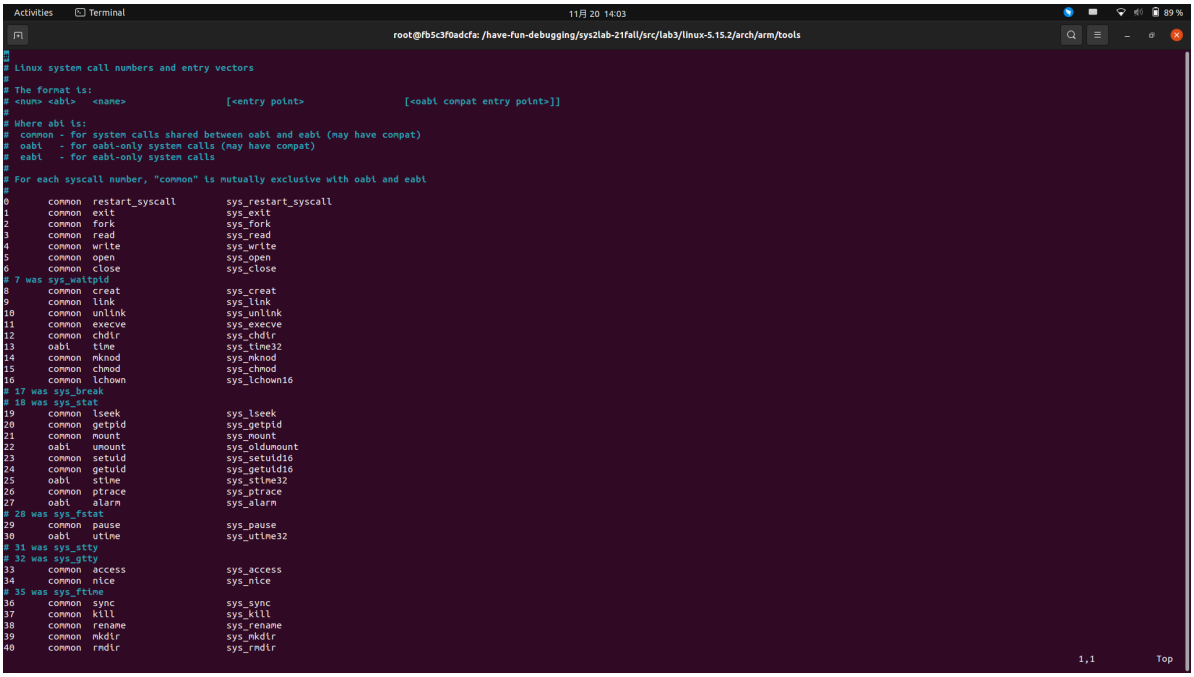

Activities
Terminal
11月 20 13:51
92 %
root@fb5c3f0adcfa: /have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86/entry/syscalls
#
32-bit system call numbers and entry vectors
#
The format is:
<number> <abi> <name> <entry point> <compat entry point>
#
The __ia32_sys and __ia32_compat_sys stubs 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
10 i386 unlink sys_unlink
11 i386 execve sys_execve compat_sys_execve
12 i386 chdir sys_chdir
13 i386 time sys_time32
14 i386 mknod sys_mknod
15 i386 chmod sys_chmod
16 i386 lchown sys_lchown16
17 i386 break
18 i386 oldstat sys_stat
19 i386 lseek sys_lseek compat_sys_lseek
20 i386 getpid sys_getpid
21 i386 mount sys_mount
22 i386 umount sys_oldumount
23 i386 setuid sys_setuid16
24 i386 getuid sys_getuid16
25 i386 stime sys_stime32
26 i386 ptrace sys_ptrace compat_sys_ptrace
27 i386 alarm sys_alarm
28 i386 oldfstat sys_fstat
29 i386 pause sys_pause
30 i386 utime sys_utime32
31 i386 stty
32 i386 gtty
33 i386 access sys_access
34 i386 nice sys_nice
35 i386 ftime
36 i386 sync sys_sync
37 i386 kill sys_kill
38 i386 rename sys_rename
39 i386 mkdir sys_mkdir
40 i386 rmdir sys_rmdir
1,1 Top

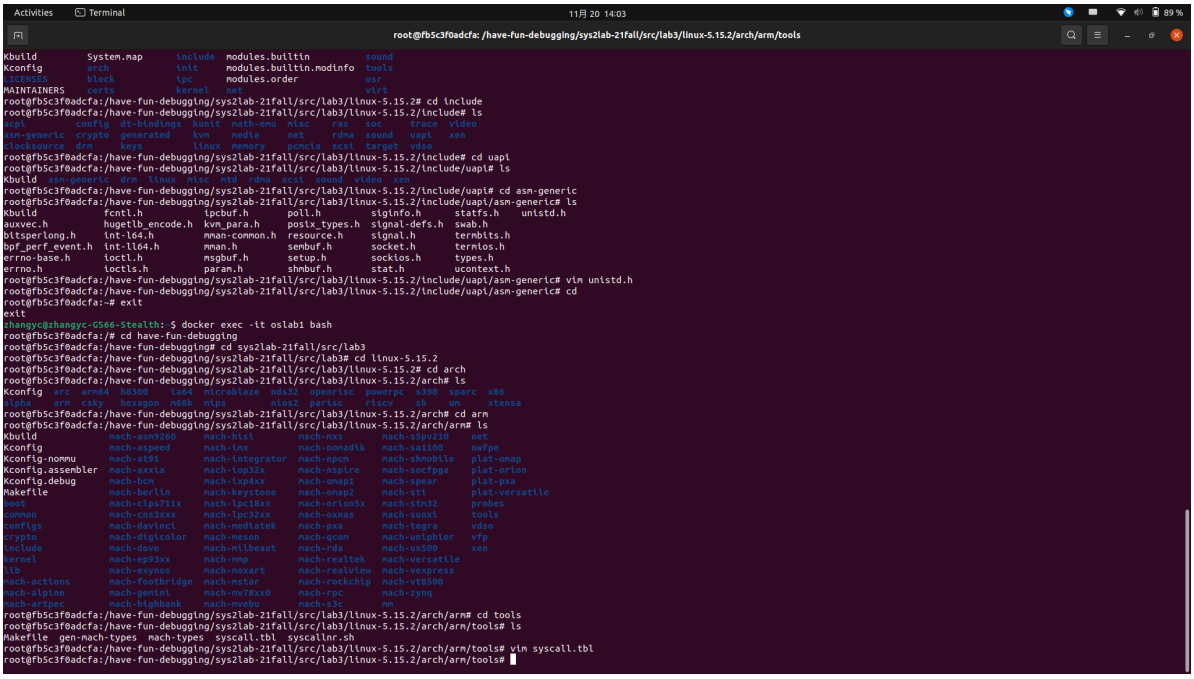

Activities
Terminal
11月 20 13:51
92 %
root@fb5c3f0adcfa: /have-fun-debugging/sys2lab-21fall/src/lab3/linux-5.15.2/arch/x86/entry/syscalls
#
64-bit system call numbers and entry vectors
#
The format is:
<number> <abi> <name> <entry point>
#
The __x64_sys_*() stubs are created on-the-fly for sys_*() system 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
13 64 rt_sigaction sys_rt_sigaction
14 common rt_sigprocmask sys_rt_sigprocmask
15 64 rt_sigreturn sys_rt_sigreturn
16 64 ioctl sys_ioctl
17 common pread64 sys_pread64
18 common pwrite64 sys_pwrite64
19 64 readv sys_readv
20 64 writev sys_writev
21 common access sys_access
22 common pipe sys_pipe
23 common select sys_select
24 common sched_yield sys_sched_yield
25 common mremap sys_mremap
26 common msync sys_msync
27 common mincore sys_mincore
28 common madvise sys_madvise
29 common shmget sys_shmget
30 common shmat sys_shmat
31 common shmctl sys_shmctl
32 common dup sys_dup
33 common dup2 sys_dup2
34 common pause sys_pause
35 common nanosleep sys_nanosleep
36 common getitimer sys_getitimer
37 common alarm sys_alarm
38 common setitimer sys_setitimer
39 common getpid sys_getpid
40 common sendfile sys_sendfile64
41 common socket sys_socket
42 common connect sys_connect
43 common accept sys_accept
"syscall_64.tbl" 417L, 14735C
1,1 Top

we can get the syscalls code on Linux for arm from arch/arm/tools



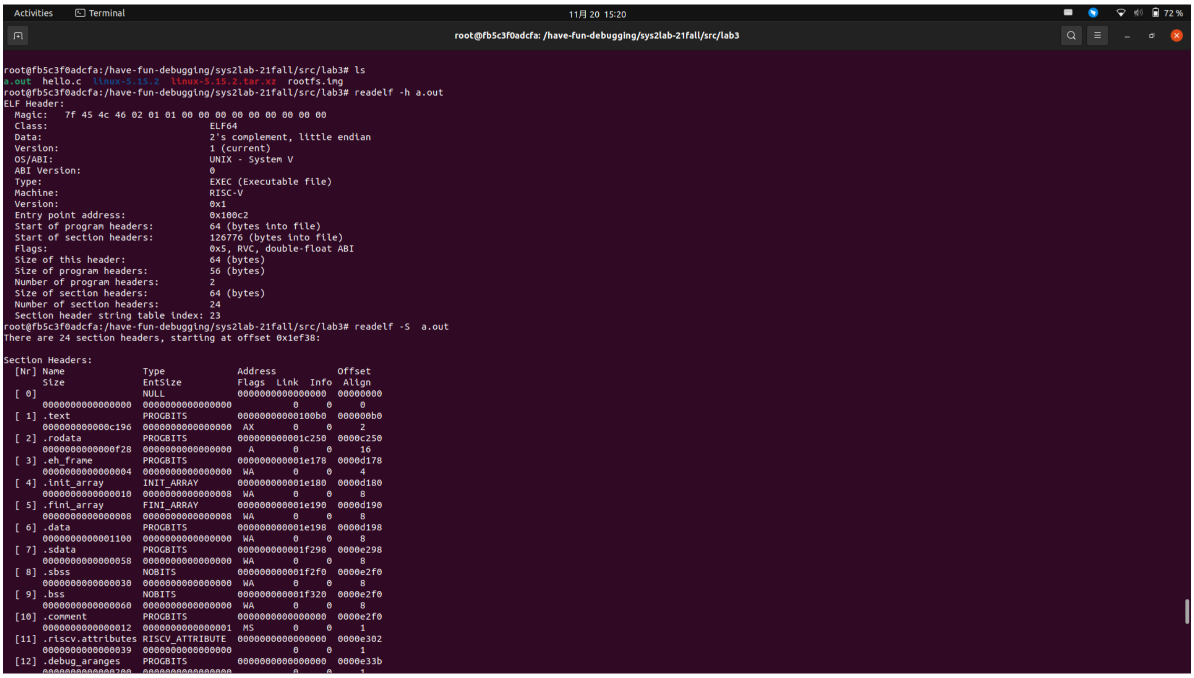






Q4

The ELF (Executable and Linkable Format),for example the exe/.so/.out/.o/core are ELF. Each ELF is include : 1.Programsm header table 2. Section header table 3. Data referred to by entries is the program header table or section header table



```
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3# ls
a.out  hello.c  linux-5.15.2  linux-5.15.2.tar.xz  rootfs.img
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3# readelf -h a.out
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           RISC-V
  Version:                           0x1
  Entry point address:               0x100c2
  Start of program headers:          64 (bytes into file)
  Start of section headers:          126776 (bytes into file)
  Flags:                             0x5, RVC, double-float ABI
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         2
  Size of section headers:           64 (bytes)
  Number of section headers:         24
  Section header string table index: 23
root@fb5c3f0adcfa:/have-fun-debugging/sys2lab-21fall/src/lab3# readelf -S  a.out
There are 24 section headers, starting at offset 0x1ef38:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         00000000000100b0  000000b0
       000000000000c196  0000000000000000  AX       0     0     2
  [ 2] .rodata           PROGBITS         000000000001c250  0000c250
       0000000000000f28  0000000000000000   A       0     0     16
  [ 3] .eh_frame         PROGBITS         000000000001e178  0000d178
       0000000000000004  0000000000000000  WA       0     0     4
  [ 4] .init_array       INIT_ARRAY       000000000001e180  0000d180
       0000000000000010  0000000000000008  WA       0     0     8
  [ 5] .fini_array       FINI_ARRAY       000000000001e190  0000d190
       0000000000000008  0000000000000008  WA       0     0     8
  [ 6] .data             PROGBITS         000000000001e198  0000d198
       0000000000001100  0000000000000000  WA       0     0     8
  [ 7] .sdata            PROGBITS         000000000001f298  0000e298
       0000000000000058  0000000000000000  WA       0     0     8
  [ 8] .sbss             NOBITS           000000000001f2f0  0000e2f0
       0000000000000030  0000000000000000  WA       0     0     8
  [ 9] .bss              NOBITS           000000000001f320  0000e2f0
       0000000000000060  0000000000000000  WA       0     0     8
  [10] .comment          PROGBITS         0000000000000000  0000e2f0
       0000000000000012  0000000000000001  MS       0     0     1
  [11] .riscv.attributes RISCV_ATTRIBUTE  0000000000000000  0000e302
       0000000000000039  0000000000000000           0     0     1
  [12] .debug_aranges    PROGBITS         0000000000000000  0000e33b
```