Homework 3

1. Operating System Concepts Chapter 1 Exercises: 1.14, 1.17, 1.19, 1.22 (20 points)

1.14 What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?

The purpose of interrupts throughout modern operating systems is to handle asynchronous events.

Differences:

1. The trap is a signal raised by a user program instructing the operating system to perform some functionality immediately. In contrast, the interrupt is a signal to the CPU emitted by hardware that indicates an event that requires immediate attention.
2. A trap also triggers OS functionality. It gives control to the trap handler. In contrast, an interrupt triggers the CPU to perform the interrupt handler routine.
3. A trap is synchronous and may occur after the execution of the instruction. In contrast, an interrupt is asynchronous and may occur at any time.
4. A trap is generated by a user program instruction. In contrast, the hardware devices generate an interrupt.

Yes, traps can be generated by a user program for instructing the operating system to do some instruction.

1.17 Some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is and that it is not possible.

If it is possible:

We can fix some system functions on the hardware and can't be rewritten. The security of the system depends on security of each application. If the programming languages are safe in memory space processing, it's possible to construct a secure operating system.

If it is not possible:

We can't promise all the applications are safe. If there isn't a privileged mode, malicious programs will rewrite the operating system code and may damage some vital functions of the operating system.

1.19 Rank the following storage systems from slowest to fastest:

a. Hard-disk drives b. Registers c. Optical disk d. Main memory e. Nonvolatile memory f. Magnetic tapes g. Cache

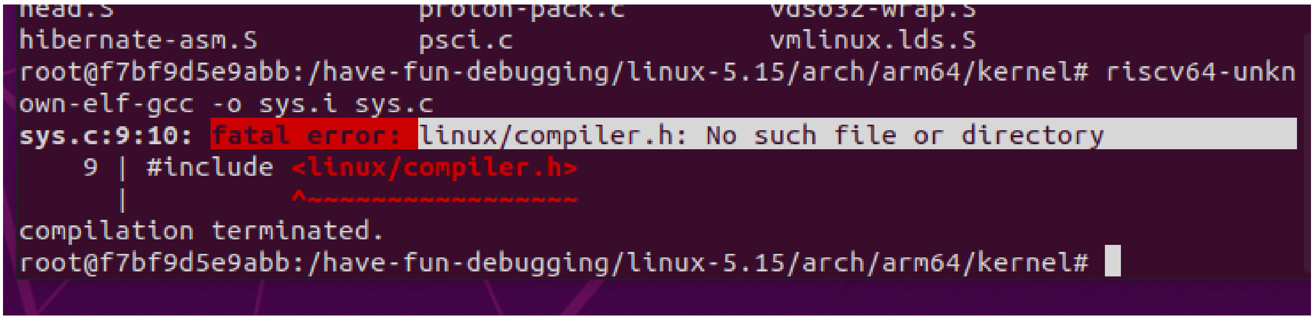
1. f. Magnetic tapes
2. c. Optical disk
3. a. Hard-disk drive
4. e. Nonvolatile memory
5. d. Main memory
6. g. Cache
7. b. Registers

1.22 Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs

Virtual memory can be used as a memory protection tool, which stores the permissions and memory addresses of each process. Kernel and user modes are set in the operating system. The system checks the mode of each process before executing the instruction. When a process attempts to perform some operations beyond its mode, it will trigger the system exception function.

2. Detail your steps about how to get arch/arm64/kernel/sys.i (10 points)

```
//启动处于停止状态的容器
$docker start oslab1  //因为我的容器名称叫oslab1
$docker ps  //查看正在运行的容器，这步用于确定容器正在运行，可以省略
$docker exec -it oslab1 bash   //将终端连入docker容器(oslab1)
root@(随机码):/#
```



```
head.S                       proton-pack.c          vdso32-wrap.S
hibernate-asm.S              psci.c                 vmlinux.lds.S
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/arch/arm64/kernel# riscv64-unkn
own-elf-gcc -o sys.i sys.c
sys.c:9:10: fatal error: linux/compiler.h: No such file or directory
    9 | #include <linux/compiler.h>
      |          ^~~~~~~~~~~~~~~~~~
compilation terminated.
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/arch/arm64/kernel# 
```

直接编译遇到错误，原因是arm架构下要用aarch tool chain。进行一下更新。

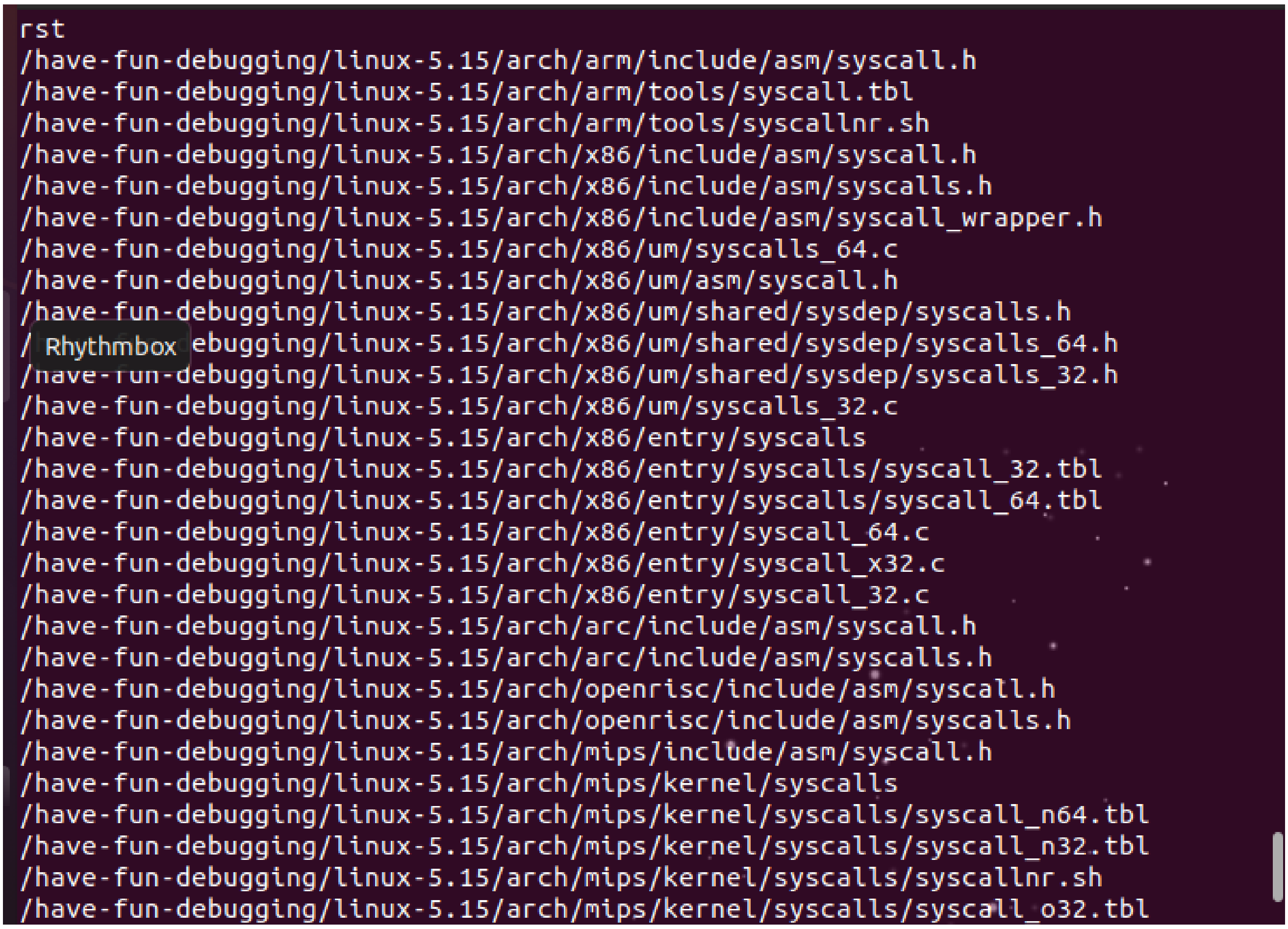
在linux-5.15文件夹路径下，依次输入以下指令

```
 #apt-get update
#apt-get install gcc-aarch64-linux-gnu
#aarch64-linux-gnu-gcc -v //查看gcc版本，如果有版本信息，这说明添加成功
#make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- defconfig//生成配置
/*这里会输出
***Default configuration is based on 'defconfig'
#
# configuration written to .config
#
以上这四行*/

#apt-get install libssl-dev
#make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- arch/arm64/kernel/sys.i -j $(nproc) //编译
//nproc内核参数,是系统上的最大进程数。使用多线程编译一般会耗费大量内存，如果 -j 选项导致内存耗尽(out of memory)，请尝试调低线程数c'd，比如 -j4, -j8 等。
```

3. Find system call table of Linux v5.15 for ARM32, RISC-V(32 bit), RISC-V(64 bit), x86(32 bit), x86_64 (50 points)

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

```
进入oslab1容器的linux-5.15文件夹
#find / -name 'syscall*' //一定要加*不然会报错
```



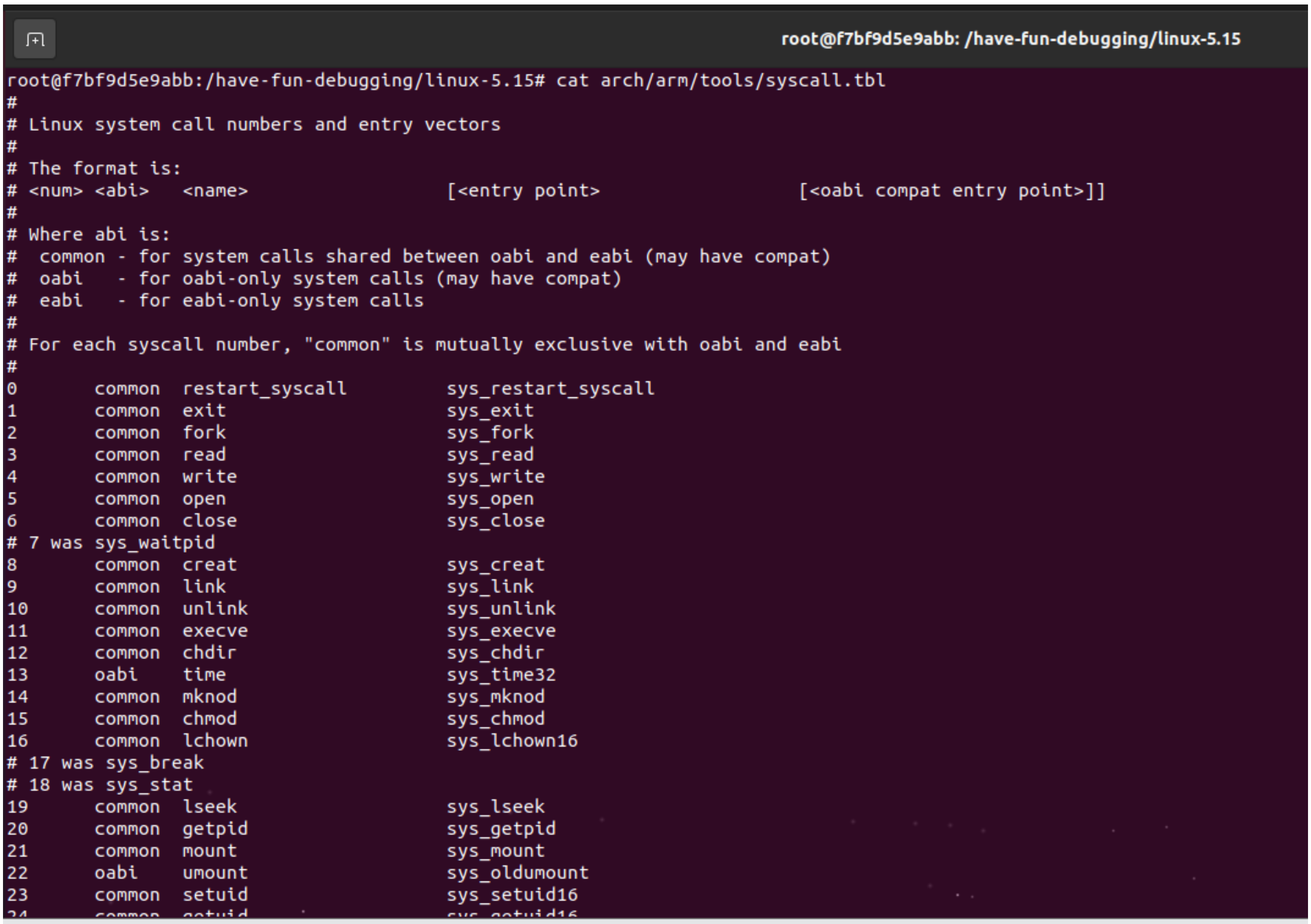
```
rst
/have-fun-debugging/linux-5.15/arch/arm/include/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/arm/tools/syscall.tbl
/have-fun-debugging/linux-5.15/arch/arm/tools/syscallnr.sh
/have-fun-debugging/linux-5.15/arch/x86/include/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/x86/include/asm/syscalls.h
/have-fun-debugging/linux-5.15/arch/x86/include/asm/syscall_wrapper.h
/have-fun-debugging/linux-5.15/arch/x86/um/syscalls_64.c
/have-fun-debugging/linux-5.15/arch/x86/um/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/x86/um/shared/sysdep/syscalls.h
/          ebugging/linux-5.15/arch/x86/um/shared/sysdep/syscalls_64.h
/have-fun-debugging/linux-5.15/arch/x86/um/shared/sysdep/syscalls_32.h
/have-fun-debugging/linux-5.15/arch/x86/um/syscalls_32.c
/have-fun-debugging/linux-5.15/arch/x86/entry/syscalls
/have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_32.tbl
/have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_64.tbl
/have-fun-debugging/linux-5.15/arch/x86/entry/syscall_64.c
/have-fun-debugging/linux-5.15/arch/x86/entry/syscall_x32.c
/have-fun-debugging/linux-5.15/arch/x86/entry/syscall_32.c
/have-fun-debugging/linux-5.15/arch/arc/include/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/arc/include/asm/syscalls.h
/have-fun-debugging/linux-5.15/arch/openrisc/include/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/openrisc/include/asm/syscalls.h
/have-fun-debugging/linux-5.15/arch/mips/include/asm/syscall.h
/have-fun-debugging/linux-5.15/arch/mips/kernel/syscalls
/have-fun-debugging/linux-5.15/arch/mips/kernel/syscalls/syscall_n64.tbl
/have-fun-debugging/linux-5.15/arch/mips/kernel/syscalls/syscall_n32.tbl
/have-fun-debugging/linux-5.15/arch/mips/kernel/syscalls/syscallnr.sh
/have-fun-debugging/linux-5.15/arch/mips/kernel/syscalls/syscall_o32.tbl
```

然后我们得到了所有文件名包含syscall的文件，在一共166个文件中我们可以发现几个有用的。

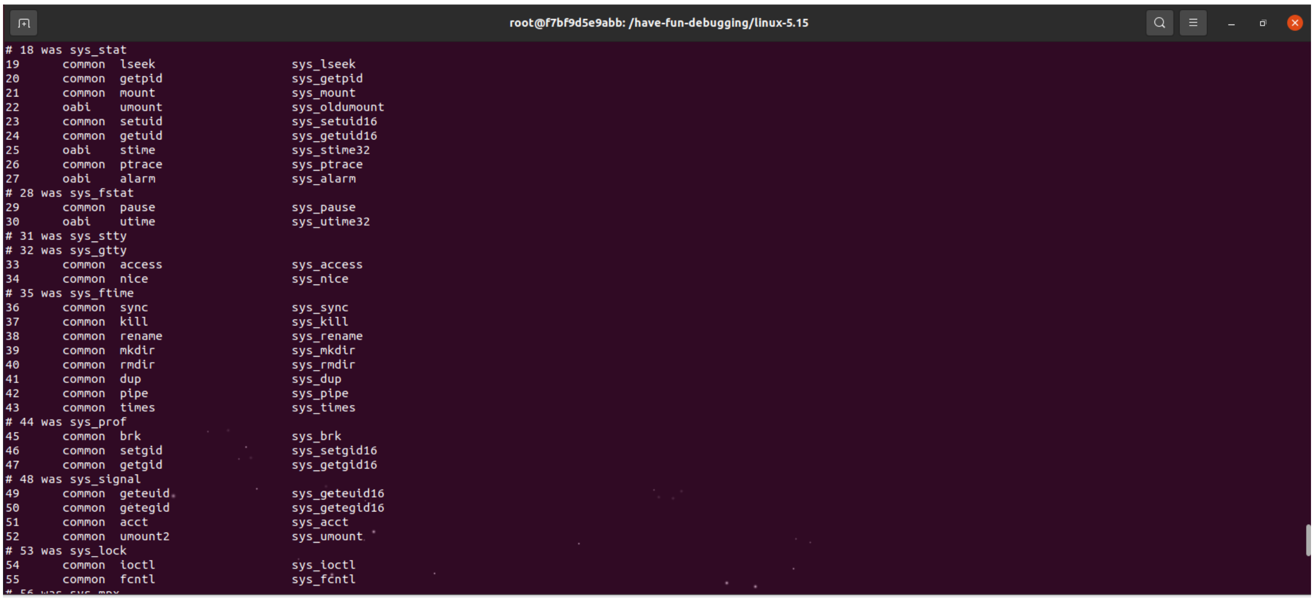
```
- /have-fun-debugging/linux-5.15/arch/riscv/kernel/syscall_table.c
- /have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_64.tbl
- /have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_32.tbl
- /have-fun-debugging/linux-5.15/arch/arm/tools/syscall.tbl
```

.tbl文件可直接在terminal里通过cat指令查看，下面三个都可以这样看到system call table，.c文件则需要通过编译获得。以下为实验截图：

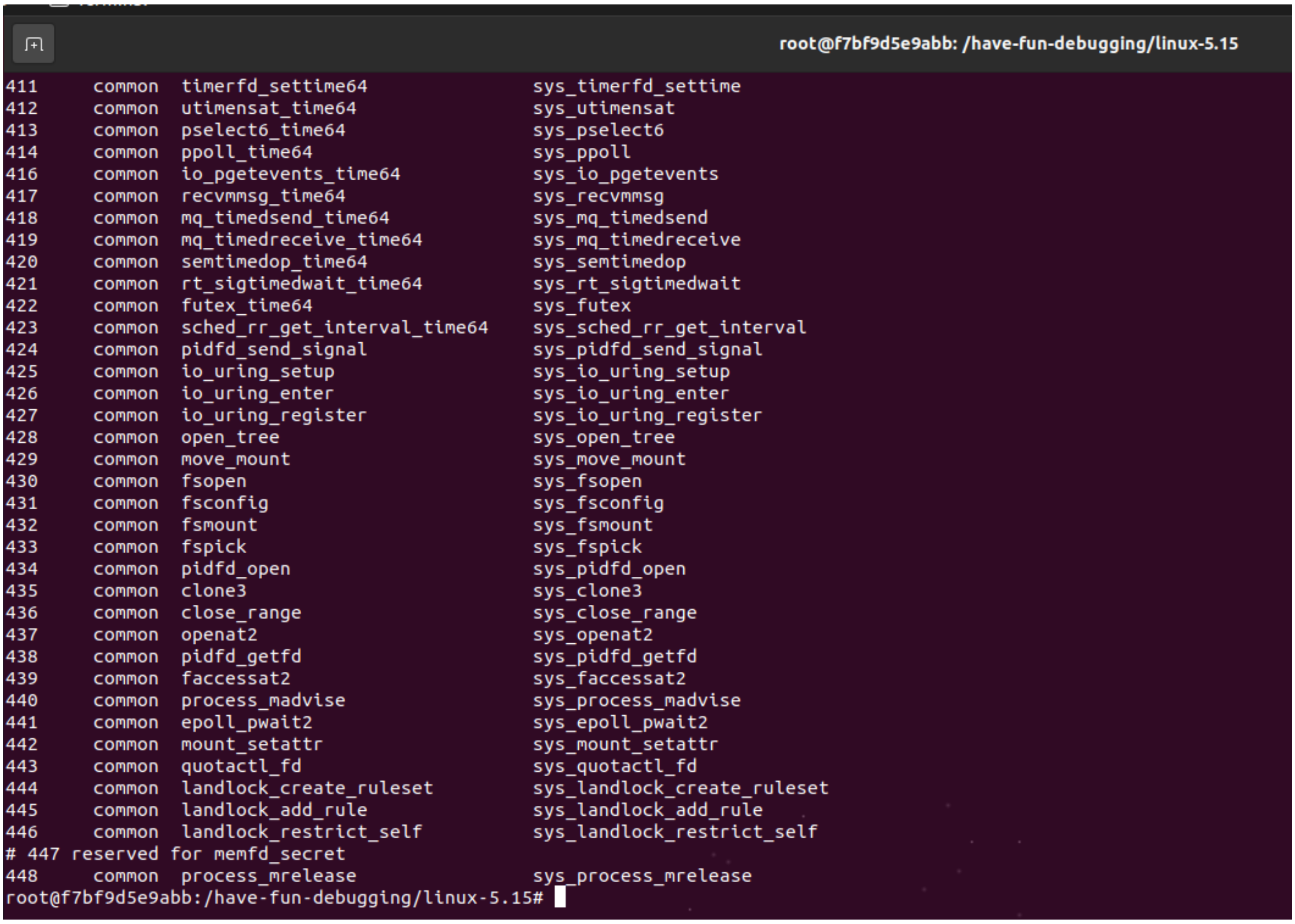
- /have-fun-debugging/linux-5.15/arch/arm/tools/syscall.tbl



```
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15# cat arch/arm/tools/syscall.tbl
#
# Linux system call numbers and entry vectors
#
# The format is:
# <num> <abi>   <name>                  [<entry point>]                 [<oabi compat entry point>]]
#
# Where abi is:
#  common - for system calls shared between oabi and eabi (may have compat)
#  oabi   - for oabi-only system calls (may have compat)
#  eabi   - for eabi-only system calls
#
# For each syscall number, "common" is mutually exclusive with oabi and eabi
#
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
23      common  setuid                  sys_setuid16
24      common  getuid                  sys_getuid16
```

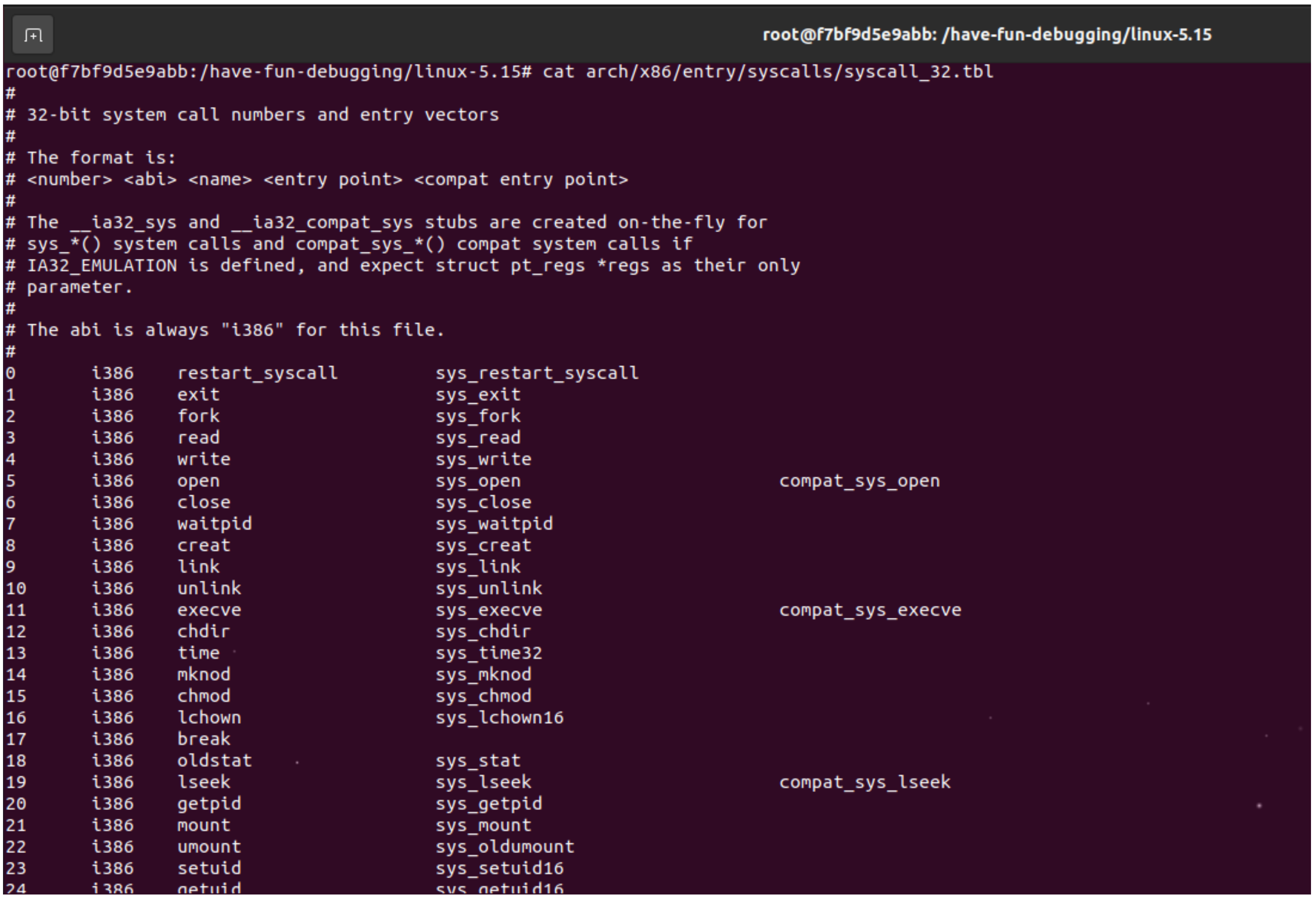


```
# 18 was sys_stat
19      common  lseek                   sys_lseek
20      common  getpid                  sys_getpid
21      common  mount                   sys_mount
22      oabi    umount                  sys_oldumount
23      common  setuid                  sys_setuid16
24      common  getuid                  sys_getuid16
25      oabi    stime                   sys_stime32
26      common  ptrace                  sys_ptrace
27      oabi    alarm                   sys_alarm
# 28 was sys_fstat
29      common  pause                   sys_pause
30      oabi    utime                   sys_utime32
# 31 was sys_stty
# 32 was sys_gtty
33      common  access                  sys_access
34      common  nice                    sys_nice
# 35 was sys_ftime
36      common  sync                    sys_sync
37      common  kill                    sys_kill
38      common  rename                  sys_rename
39      common  mkdir                   sys_mkdir
40      common  rmdir                   sys_rmdir
41      common  dup                     sys_dup
42      common  pipe                    sys_pipe
43      common  times                   sys_times
# 44 was sys_prof
45      common  brk                     sys_brk
46      common  setgid                  sys_setgid16
47      common  getgid                  sys_getgid16
# 48 was sys_signal
49      common  geteuid                 sys_geteuid16
50      common  getegid                 sys_getegid16
51      common  acct                    sys_acct
52      common  umount2                 sys_umount
# 53 was sys_lock
54      common  ioctl                   sys_ioctl
55      common  fcntl                   sys_fcntl
# 56 was sys_mpx
```

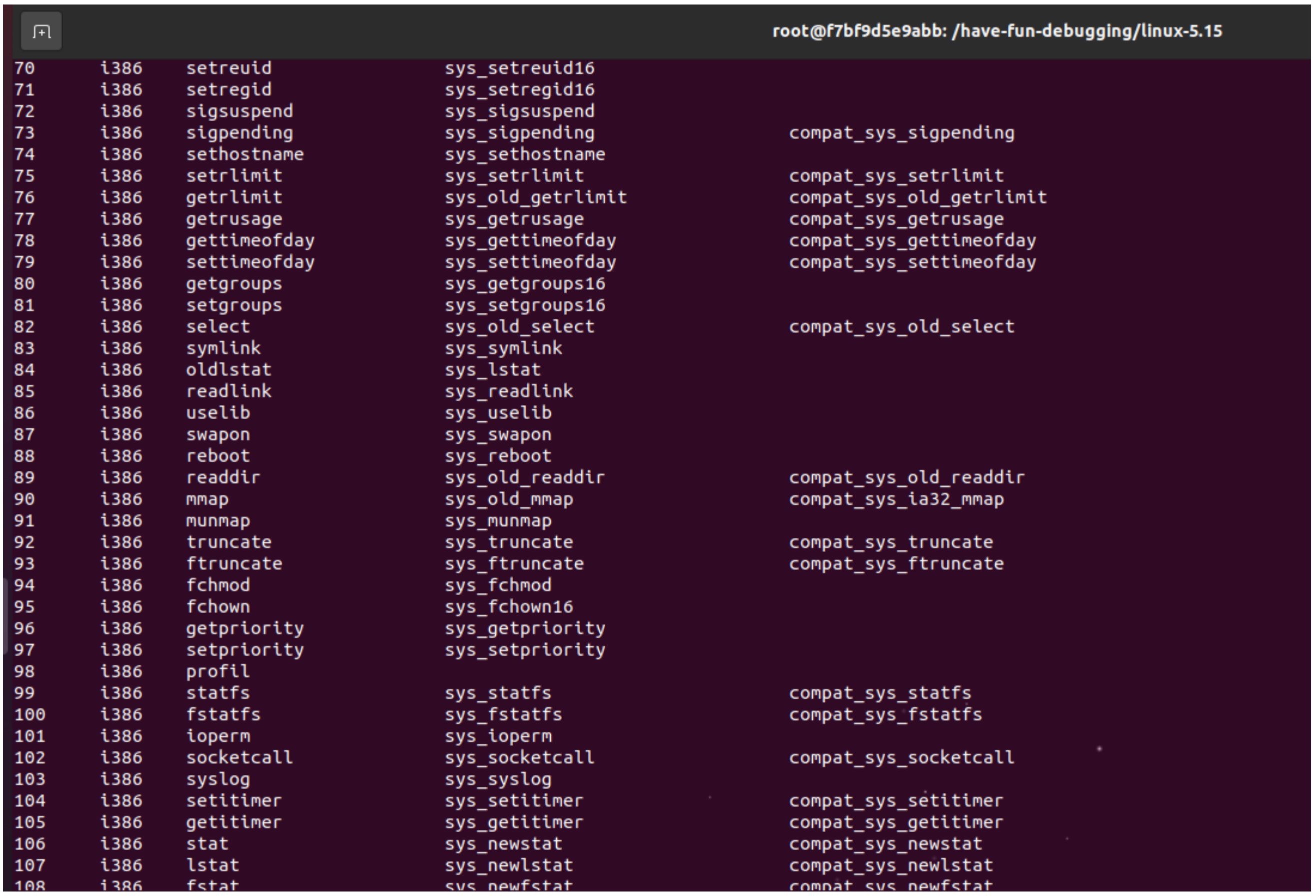


```
411     common  timerfd_settime64               sys_timerfd_settime
412     common  utimensat_time64                sys_utimensat
413     common  pselect6_time64                 sys_pselect6
414     common  ppoll_time64                    sys_ppoll
416     common  io_pgetevents_time64            sys_io_pgetevents
417     common  recvmmsg_time64                 sys_recvmmsg
418     common  mq_timedsend_time64             sys_mq_timedsend
419     common  mq_timedreceive_time64          sys_mq_timedreceive
420     common  semtimedop_time64               sys_semtimedop
421     common  rt_sigtimedwait_time64          sys_rt_sigtimedwait
422     common  futex_time64                    sys_futex
423     common  sched_rr_get_interval_time64    sys_sched_rr_get_interval
424     common  pidfd_send_signal               sys_pidfd_send_signal
425     common  io_uring_setup                  sys_io_uring_setup
426     common  io_uring_enter                  sys_io_uring_enter
427     common  io_uring_register               sys_io_uring_register
428     common  open_tree                       sys_open_tree
429     common  move_mount                      sys_move_mount
430     common  fsopen                          sys_fsopen
431     common  fsconfig                        sys_fsconfig
432     common  fsmount                         sys_fsmount
433     common  fspick                          sys_fspick
434     common  pidfd_open                      sys_pidfd_open
435     common  clone3                          sys_clone3
436     common  close_range                     sys_close_range
437     common  openat2                         sys_openat2
438     common  pidfd_getfd                     sys_pidfd_getfd
439     common  faccessat2                      sys_faccessat2
440     common  process_madvise                 sys_process_madvise
441     common  epoll_pwait2                    sys_epoll_pwait2
442     common  mount_setattr                   sys_mount_setattr
443     common  quotactl_fd                     sys_quotactl_fd
444     common  landlock_create_ruleset         sys_landlock_create_ruleset
445     common  landlock_add_rule               sys_landlock_add_rule
446     common  landlock_restrict_self          sys_landlock_restrict_self
# 447 reserved for memfd_secret
448     common  process_mrelease                sys_process_mrelease
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15#
```

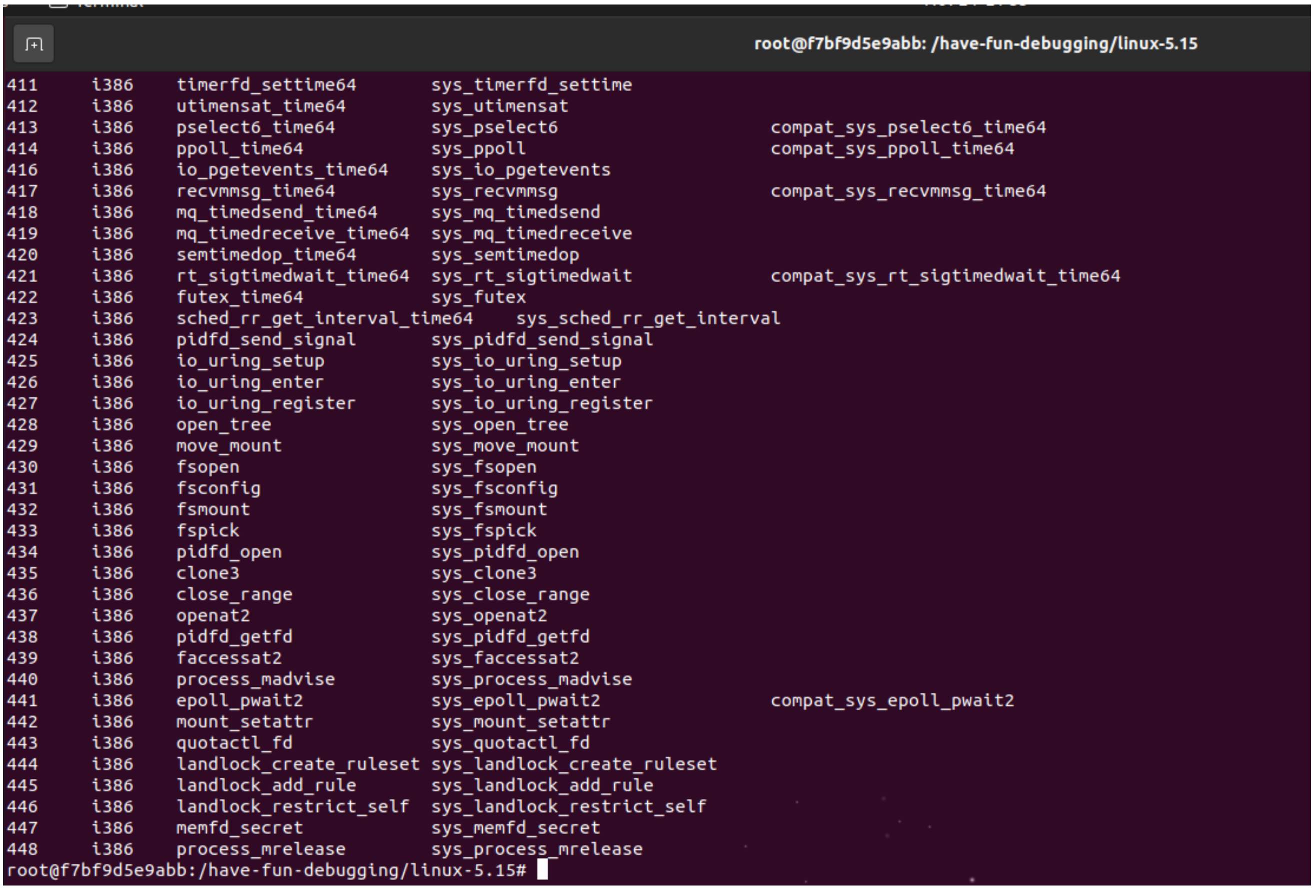
- /have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_32.tbl



```
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15# cat arch/x86/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
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
```



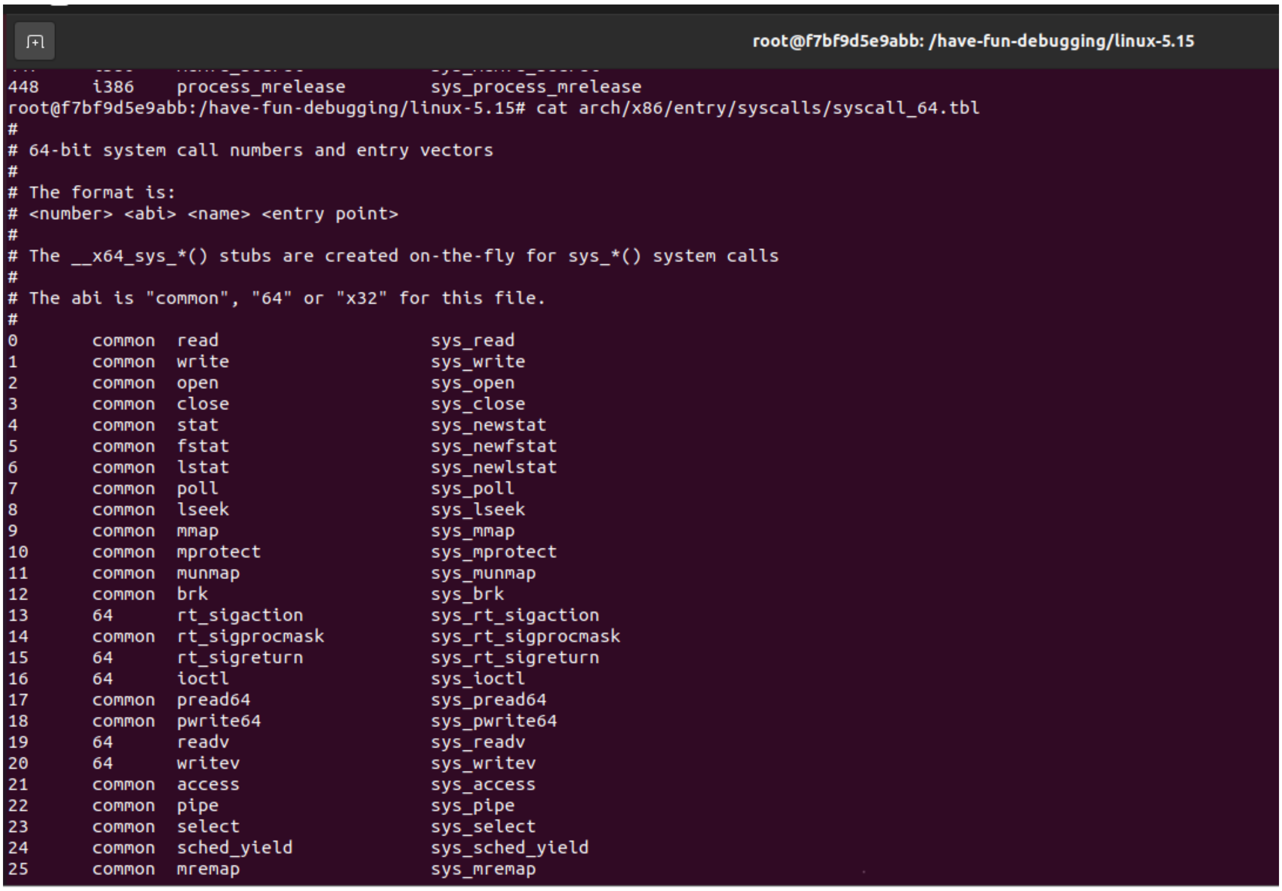
```
70      i386    setreuid                sys_setreuid16
71      i386    setregid                sys_setregid16
72      i386    sigsuspend              sys_sigsuspend
73      i386    sigpending              sys_sigpending                  compat_sys_sigpending
74      i386    sethostname             sys_sethostname
75      i386    setrlimit               sys_setrlimit                   compat_sys_setrlimit
76      i386    getrlimit               sys_old_getrlimit               compat_sys_old_getrlimit
77      i386    getrusage               sys_getrusage                   compat_sys_getrusage
78      i386    gettimeofday            sys_gettimeofday                compat_sys_gettimeofday
79      i386    settimeofday            sys_settimeofday                compat_sys_settimeofday
80      i386    getgroups               sys_getgroups16
81      i386    setgroups               sys_setgroups16
82      i386    select                  sys_old_select                  compat_sys_old_select
83      i386    symlink                 sys_symlink
84      i386    oldlstat                sys_lstat
85      i386    readlink                sys_readlink
86      i386    uselib                  sys_uselib
87      i386    swapon                  sys_swapon
88      i386    reboot                  sys_reboot
89      i386    readdir                 sys_old_readdir                 compat_sys_old_readdir
90      i386    mmap                    sys_old_mmap                    compat_sys_ia32_mmap
91      i386    munmap                  sys_munmap
92      i386    truncate                sys_truncate                    compat_sys_truncate
93      i386    ftruncate               sys_ftruncate                   compat_sys_ftruncate
94      i386    fchmod                  sys_fchmod
95      i386    fchown                  sys_fchown16
96      i386    getpriority             sys_getpriority
97      i386    setpriority             sys_setpriority
98      i386    profil
99      i386    statfs                  sys_statfs                      compat_sys_statfs
100     i386    fstatfs                 sys_fstatfs                     compat_sys_fstatfs
101     i386    ioperm                  sys_ioperm
102     i386    socketcall              sys_socketcall                  compat_sys_socketcall
103     i386    syslog                  sys_syslog
104     i386    setitimer               sys_setitimer                   compat_sys_setitimer
105     i386    getitimer               sys_getitimer                   compat_sys_getitimer
106     i386    stat                    sys_newstat                     compat_sys_newstat
107     i386    lstat                   sys_newlstat                    compat_sys_newlstat
108     i386    fstat                   sys_newfstat                    compat_sys_newfstat
```



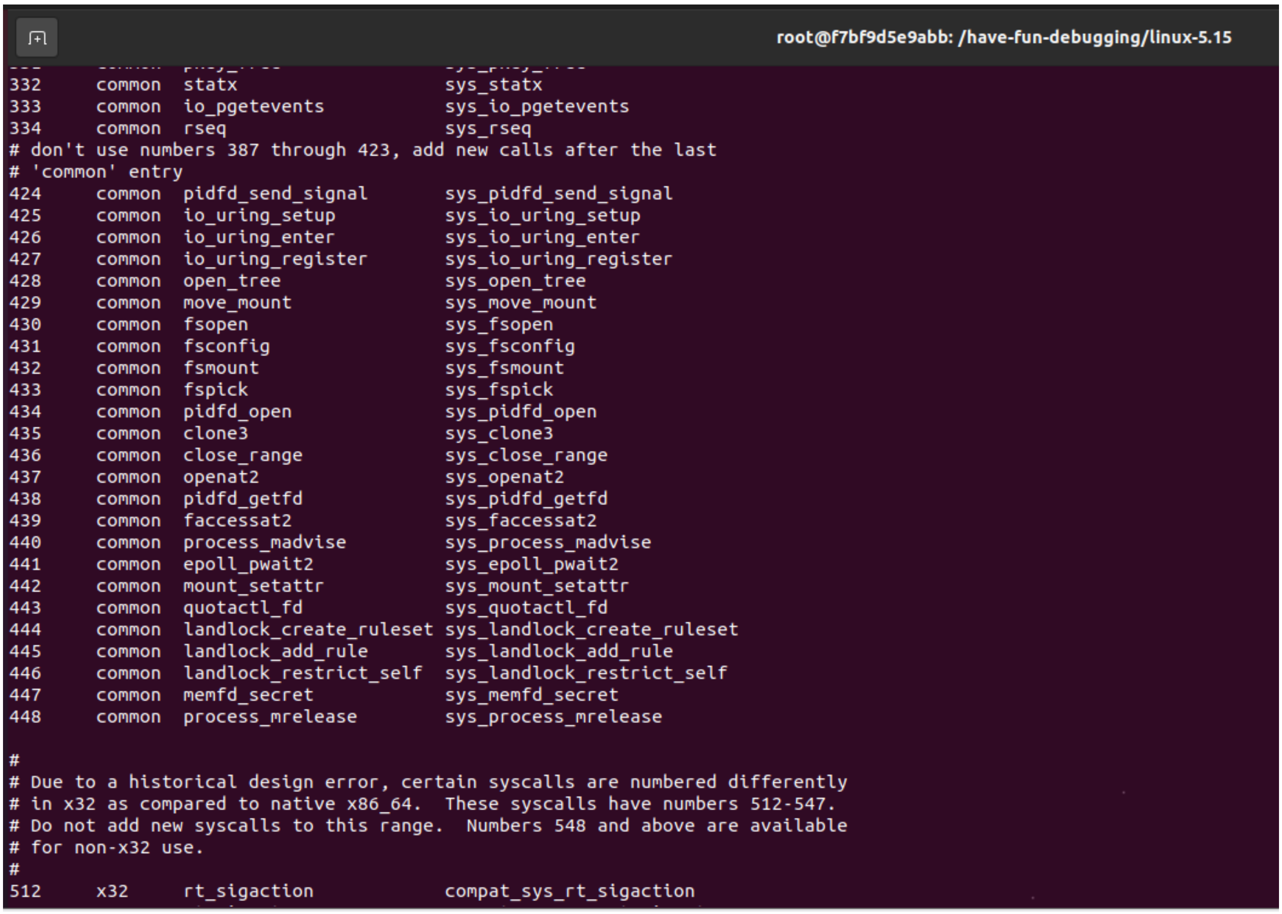
```
411     i386    timerfd_settime64       sys_timerfd_settime
412     i386    utimensat_time64        sys_utimensat
413     i386    pselect6_time64         sys_pselect6                    compat_sys_pselect6_time64
414     i386    ppoll_time64            sys_ppoll                       compat_sys_ppoll_time64
416     i386    io_pgetevents_time64    sys_io_pgetevents
417     i386    recvmmsg_time64         sys_recvmmsg                    compat_sys_recvmmsg_time64
418     i386    mq_timedsend_time64     sys_mq_timedsend
419     i386    mq_timedreceive_time64  sys_mq_timedreceive
420     i386    semtimedop_time64       sys_semtimedop
421     i386    rt_sigtimedwait_time64  sys_rt_sigtimedwait             compat_sys_rt_sigtimedwait_time64
422     i386    futex_time64            sys_futex
423     i386    sched_rr_get_interval_time64    sys_sched_rr_get_interval
424     i386    pidfd_send_signal       sys_pidfd_send_signal
425     i386    io_uring_setup          sys_io_uring_setup
426     i386    io_uring_enter          sys_io_uring_enter
427     i386    io_uring_register       sys_io_uring_register
428     i386    open_tree               sys_open_tree
429     i386    move_mount              sys_move_mount
430     i386    fsopen                  sys_fsopen
431     i386    fsconfig                sys_fsconfig
432     i386    fsmount                 sys_fsmount
433     i386    fspick                  sys_fspick
434     i386    pidfd_open              sys_pidfd_open
435     i386    clone3                  sys_clone3
436     i386    close_range             sys_close_range
437     i386    openat2                 sys_openat2
438     i386    pidfd_getfd             sys_pidfd_getfd
439     i386    faccessat2              sys_faccessat2
440     i386    process_madvise         sys_process_madvise
441     i386    epoll_pwait2            sys_epoll_pwait2                compat_sys_epoll_pwait2
442     i386    mount_setattr           sys_mount_setattr
443     i386    quotactl_fd             sys_quotactl_fd
444     i386    landlock_create_ruleset sys_landlock_create_ruleset
445     i386    landlock_add_rule       sys_landlock_add_rule
446     i386    landlock_restrict_self  sys_landlock_restrict_self
447     i386    memfd_secret            sys_memfd_secret
448     i386    process_mrelease        sys_process_mrelease
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15#
```

共计448个系统调用函数表

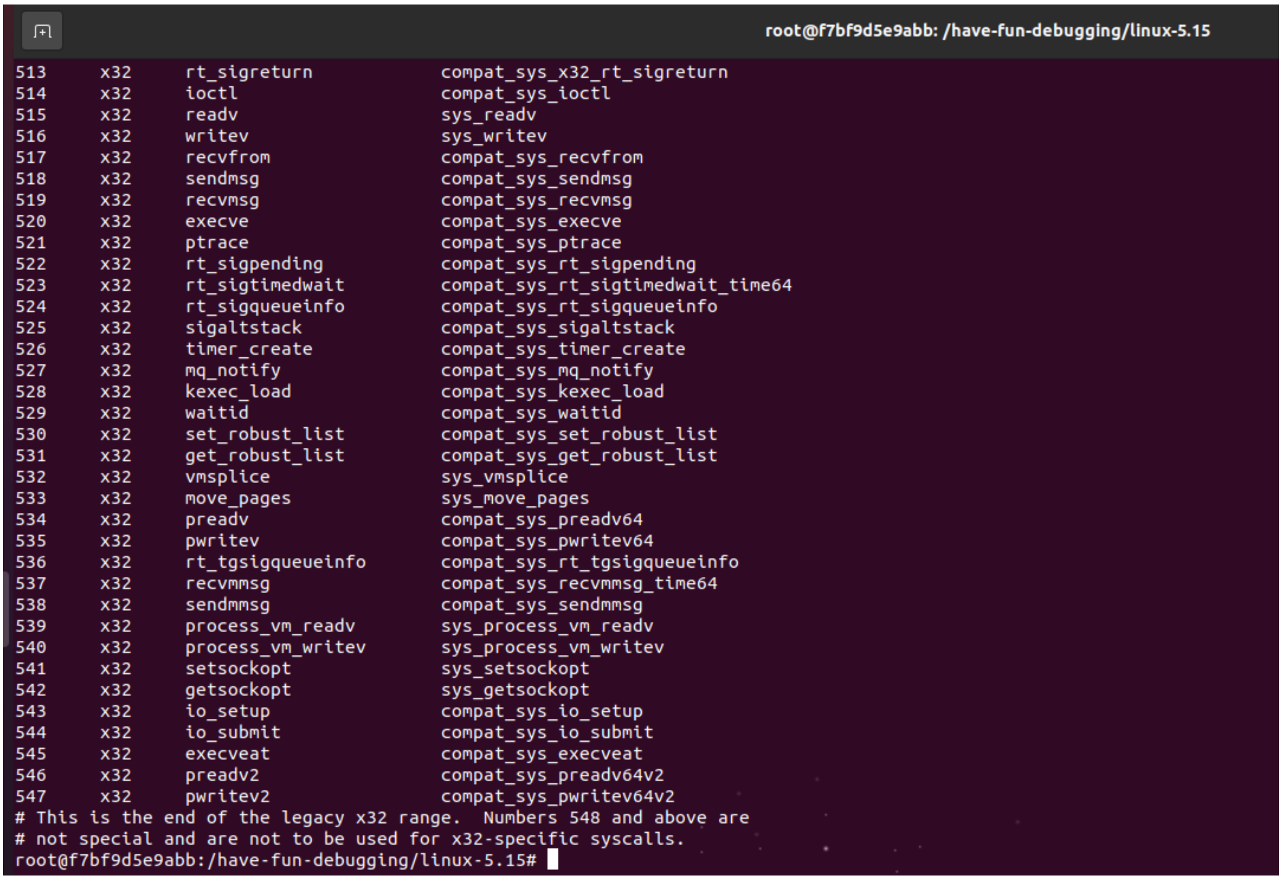
- /have-fun-debugging/linux-5.15/arch/x86/entry/syscalls/syscall_64.tbl



```
448     i386    process_mrelease        sys_process_mrelease
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15# cat arch/x86/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
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
```



```
332     common  statx                   sys_statx
333     common  io_pgetevents           sys_io_pgetevents
334     common  rseq                    sys_rseq
# don't use numbers 387 through 423, add new calls after the last
# 'common' entry
424     common  pidfd_send_signal       sys_pidfd_send_signal
425     common  io_uring_setup          sys_io_uring_setup
426     common  io_uring_enter          sys_io_uring_enter
427     common  io_uring_register       sys_io_uring_register
428     common  open_tree               sys_open_tree
429     common  move_mount              sys_move_mount
430     common  fsopen                  sys_fsopen
431     common  fsconfig                sys_fsconfig
432     common  fsmount                 sys_fsmount
433     common  fspick                  sys_fspick
434     common  pidfd_open              sys_pidfd_open
435     common  clone3                  sys_clone3
436     common  close_range             sys_close_range
437     common  openat2                 sys_openat2
438     common  pidfd_getfd             sys_pidfd_getfd
439     common  faccessat2              sys_faccessat2
440     common  process_madvise         sys_process_madvise
441     common  epoll_pwait2            sys_epoll_pwait2
442     common  mount_setattr           sys_mount_setattr
443     common  quotactl_fd             sys_quotactl_fd
444     common  landlock_create_ruleset sys_landlock_create_ruleset
445     common  landlock_add_rule       sys_landlock_add_rule
446     common  landlock_restrict_self  sys_landlock_restrict_self
447     common  memfd_secret            sys_memfd_secret
448     common  process_mrelease        sys_process_mrelease

#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64.  These syscalls have numbers 512-547.
# Do not add new syscalls to this range.  Numbers 548 and above are available
# for non-x32 use.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
```



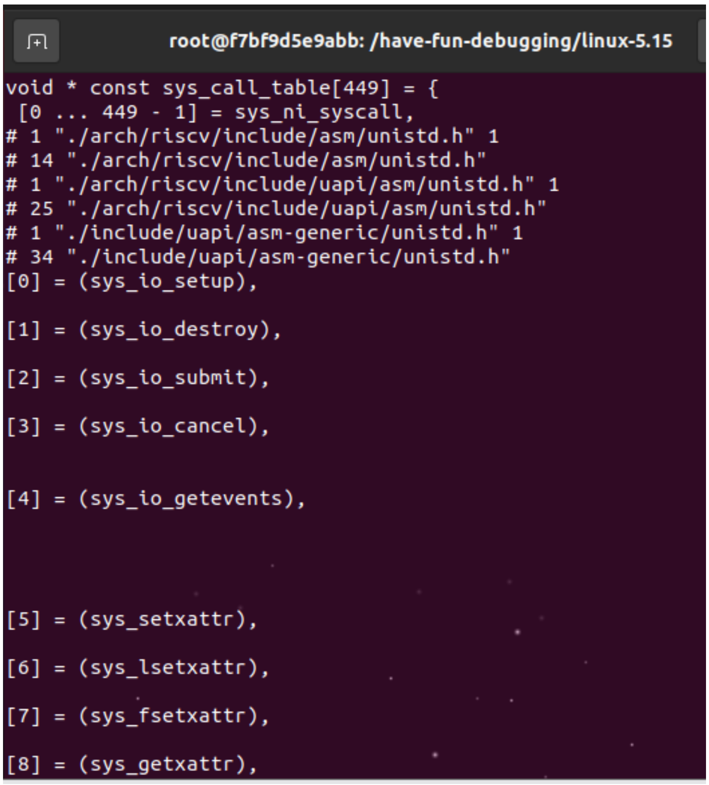
```
513     x32     rt_sigreturn            compat_sys_x32_rt_sigreturn
514     x32     ioctl                   compat_sys_ioctl
515     x32     readv                   sys_readv
516     x32     writev                  sys_writev
517     x32     recvfrom                compat_sys_recvfrom
518     x32     sendmsg                 compat_sys_sendmsg
519     x32     recvmsg                 compat_sys_recvmsg
520     x32     execve                  compat_sys_execve
521     x32     ptrace                  compat_sys_ptrace
522     x32     rt_sigpending           compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         compat_sys_rt_sigtimedwait_time64
524     x32     rt_sigqueueinfo         compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             compat_sys_sigaltstack
526     x32     timer_create            compat_sys_timer_create
527     x32     mq_notify               compat_sys_mq_notify
528     x32     kexec_load              compat_sys_kexec_load
529     x32     waitid                  compat_sys_waitid
530     x32     set_robust_list         compat_sys_set_robust_list
531     x32     get_robust_list         compat_sys_get_robust_list
532     x32     vmsplice                sys_vmsplice
533     x32     move_pages              sys_move_pages
534     x32     preadv                  compat_sys_preadv64
535     x32     pwritev                 compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                compat_sys_recvmmsg_time64
538     x32     sendmmsg                compat_sys_sendmmsg
539     x32     process_vm_readv        sys_process_vm_readv
540     x32     process_vm_writev       sys_process_vm_writev
541     x32     setsockopt              sys_setsockopt
542     x32     getsockopt              sys_getsockopt
543     x32     io_setup                compat_sys_io_setup
544     x32     io_submit               compat_sys_io_submit
545     x32     execveat                compat_sys_execveat
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
# This is the end of the legacy x32 range.  Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15#
```

共计547个系统调用函数

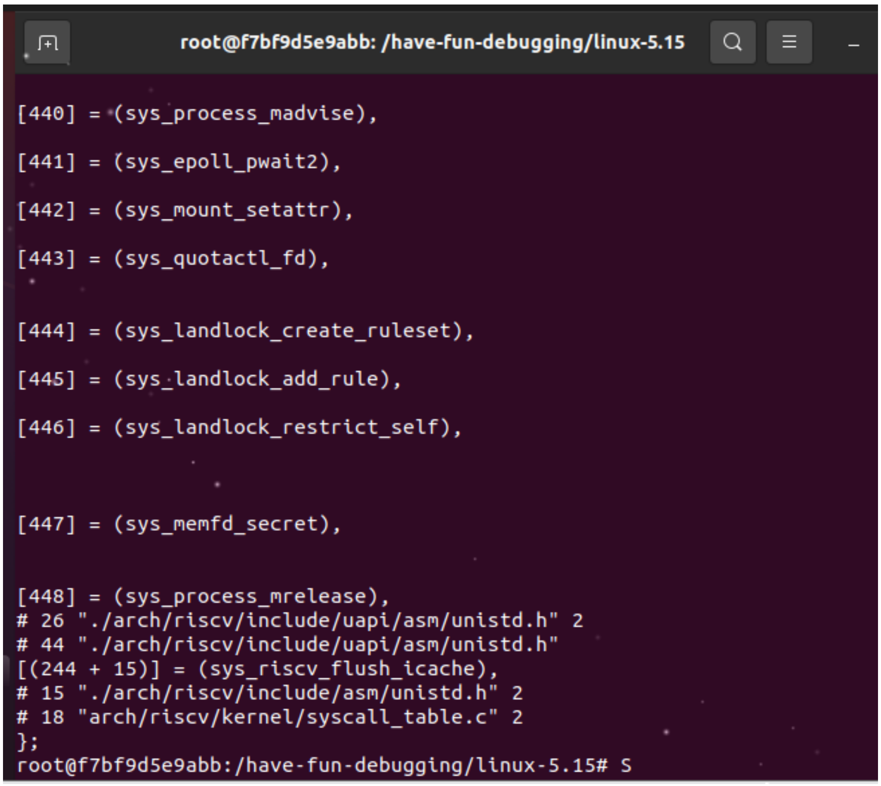
- /have-fun-debugging/linux-5.15/arch/riscv/kernel/syscall_table.c

在riscv文件夹中，只有这一个syscall_table.c函数记录了系统函数调用表。在64位和32位不同编译器中编出的汇编语言可能不一样，但其函数调用表的序号和函数是一样的，故在此只展示出64位riscv64-unknown-linux-gnu-gcc编译出的系统调用表。

```
#make ARCH=riscv CROSS_COMPILE=riscv64-unknown-linux-gnu- arch/riscv/kernel/syscall_table.i -j$(nproc)
这里要注意中间是.i不是.c，中间是目标文件
```



```
void * const sys_call_table[449] = {
 [0 ... 449 - 1] = sys_ni_syscall,
# 1 "./arch/riscv/include/asm/unistd.h" 1
# 14 "./arch/riscv/include/asm/unistd.h"
# 1 "./arch/riscv/include/uapi/asm/unistd.h" 1
# 25 "./arch/riscv/include/uapi/asm/unistd.h"
# 1 "./include/uapi/asm-generic/unistd.h" 1
# 34 "./include/uapi/asm-generic/unistd.h"
[0] = (sys_io_setup),

[1] = (sys_io_destroy),

[2] = (sys_io_submit),

[3] = (sys_io_cancel),


[4] = (sys_io_getevents),




[5] = (sys_setxattr),

[6] = (sys_lsetxattr),

[7] = (sys_fsetxattr),

[8] = (sys_getxattr),
```



```
[440] = (sys_process_madvise),

[441] = (sys_epoll_pwait2),

[442] = (sys_mount_setattr),

[443] = (sys_quotactl_fd),


[444] = (sys_landlock_create_ruleset),

[445] = (sys_landlock_add_rule),

[446] = (sys_landlock_restrict_self),



[447] = (sys_memfd_secret),


[448] = (sys_process_mrelease),
# 26 "./arch/riscv/include/uapi/asm/unistd.h" 2
# 44 "./arch/riscv/include/uapi/asm/unistd.h"
[(244 + 15)] = (sys_riscv_flush_icache),
# 15 "./arch/riscv/include/asm/unistd.h" 2
# 18 "arch/riscv/kernel/syscall_table.c" 2
};
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15# S
```

4.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.

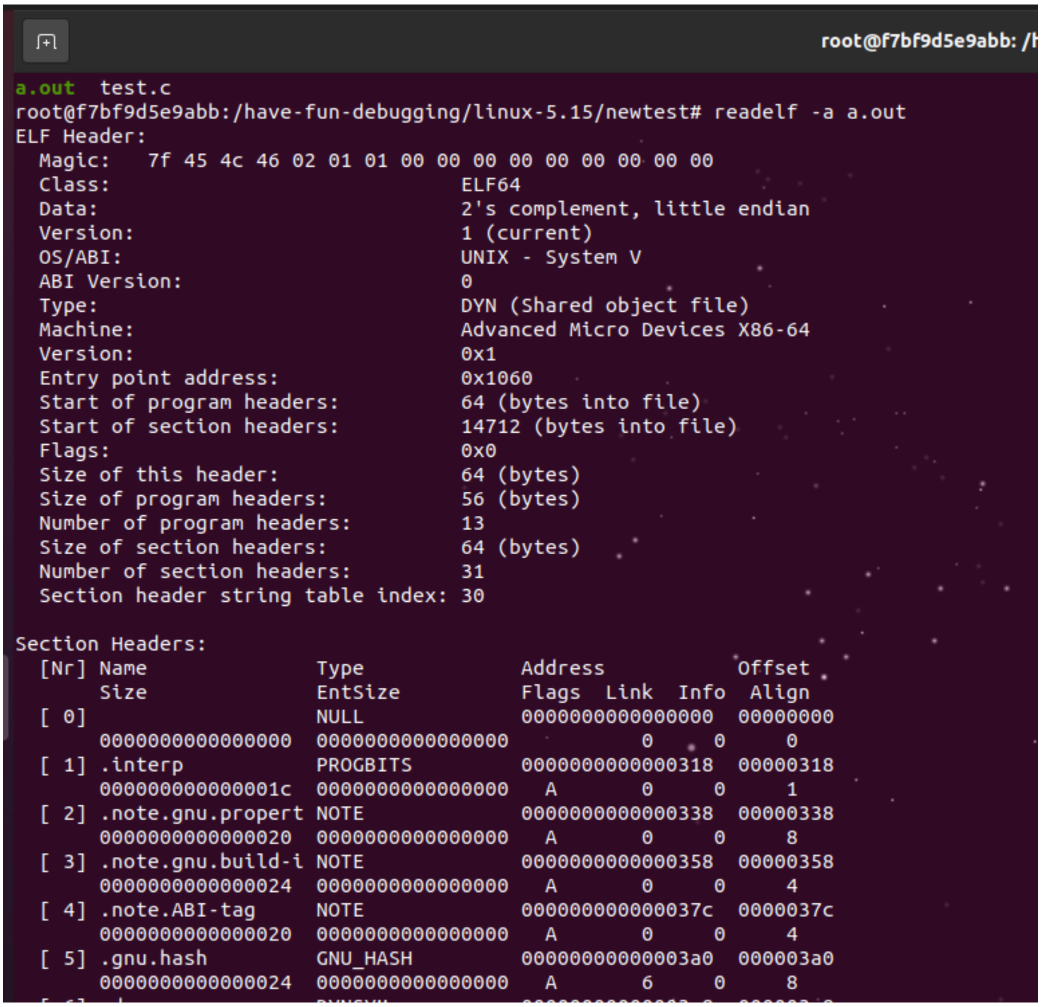
Executable and Linkable Format. It is a common standard file format for executable files, object code, shared libraries, and core dumps.

```
#touch test.c  //新建test.c
#vi test.c  //进入编辑test.c
Esc + :exit  //退出编辑
#gcc test.c  //没有规定文件名称，默认生成了a.out
#readelf -a a.out //all显示全部信息，等价于-h-l-S-s-r-d-V-A-I
```

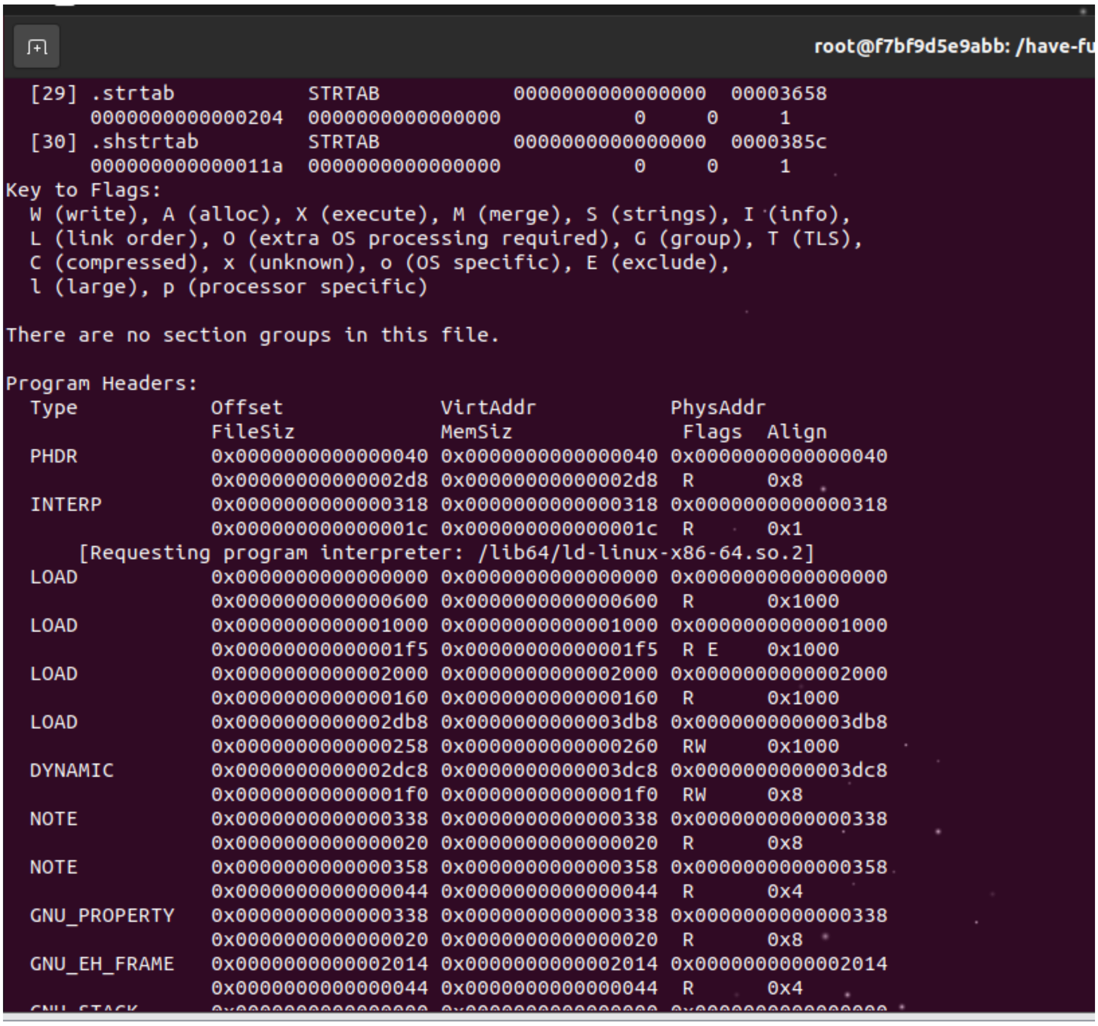
readelf 只能看elf文件的信息

- **选项-h(elfheader)，显示elf文件开始的文件头信息。后面文章会补上具体说明。**
- **选项-l(programheaders)，segments显示程序头（段头）信息(如果有数据的话)。后面文章会补上具体说明。**
- **选项-S(sectionheaders)，sections显示节头信息(如果有数据的话)。后面文章会补上具体说明。**
- 选项-a，all显示全部信息，等价于-h-l-S-s-r-d-V-A-I。
- 选项-g(sectiongroups)，显示节组信息(如果有数据的话)。
- 选项-t，section-details显示节的详细信息(-S的)。
- 选项-s，symbols显示符号表段中的项（如果有数据的话）。
- 选项-e，headers显示全部头信息，等价于:-h-l-S。
- 选项-n，notes显示note段（内核注释）的信息。
- 选项-r，relocs显示可重定位段的信息。
- 选项-u，unwind显示unwind段信息。当前只支持IA64ELF的unwind段信息。
- 选项-d，dynamic显示动态段的信息。
- 选项-V，version-info显示版本段的信息。
- 选项-A，arch-specific显示CPU构架信息。
- 选项-I，histogram显示符号的时候，显示bucketlist长度的柱状图。
- 选项-x,hex-dump=以16进制方式显示指定段内内容。number指定段表中段的索引，或字符串指定文件中的段名
- 选项-D，use-dynamic使用动态段中的符号表显示符号，而不是使用符号段。

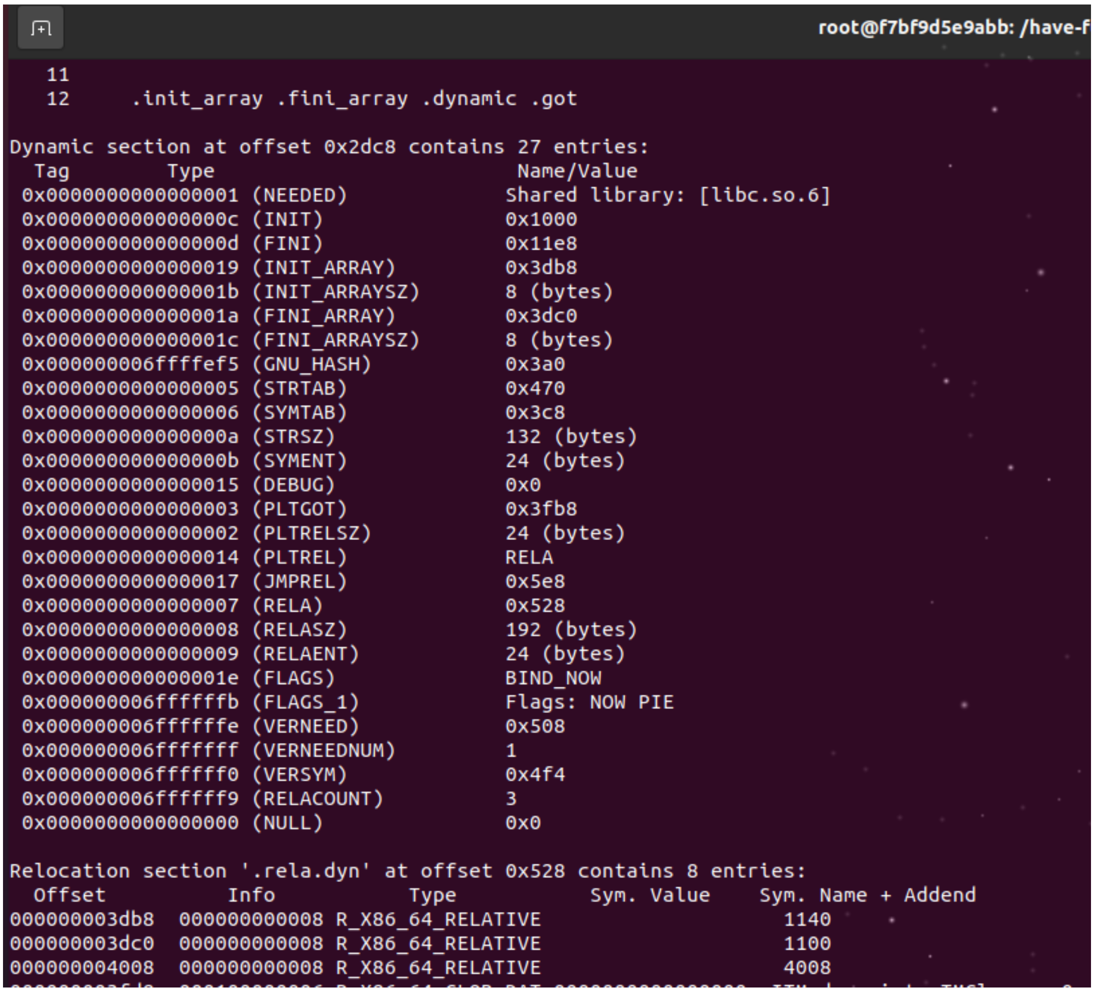
- 选项-v，version显示readelf的版本信息。
- 选项-H，help显示readelf所支持的命令行选项。



```
a.out  test.c
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# readelf -a 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:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1060
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14712 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS         0000000000000318  00000318
       000000000000001c  0000000000000000   A       0     0     1
  [ 2] .note.gnu.propert NOTE             0000000000000338  00000338
       0000000000000020  0000000000000000   A       0     0     8
  [ 3] .note.gnu.build-i NOTE             0000000000000358  00000358
       0000000000000024  0000000000000000   A       0     0     4
  [ 4] .note.ABI-tag     NOTE             000000000000037c  0000037c
       0000000000000020  0000000000000000   A       0     0     4
  [ 5] .gnu.hash         GNU_HASH         00000000000003a0  000003a0
       0000000000000024  0000000000000000   A       6     0     8
```



```
  [29] .strtab           STRTAB           0000000000000000  00003658
       0000000000000204  0000000000000000           0     0     1
  [30] .shstrtab         STRTAB           0000000000000000  0000385c
       000000000000011a  0000000000000000           0     0     1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  l (large), p (processor specific)

There are no section groups in this file.

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000000040 0x0000000000000040
                 0x00000000000002d8 0x00000000000002d8  R      0x8
  INTERP         0x0000000000000318 0x0000000000000318 0x0000000000000318
                 0x000000000000001c 0x000000000000001c  R      0x1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000600 0x0000000000000600  R      0x1000
  LOAD           0x0000000000001000 0x0000000000001000 0x0000000000001000
                 0x00000000000001f5 0x00000000000001f5  R E    0x1000
  LOAD           0x0000000000002000 0x0000000000002000 0x0000000000002000
                 0x0000000000000160 0x0000000000000160  R      0x1000
  LOAD           0x0000000000002db8 0x0000000000003db8 0x0000000000003db8
                 0x0000000000000258 0x0000000000000260  RW     0x1000
  DYNAMIC        0x0000000000002dc8 0x0000000000003dc8 0x0000000000003dc8
                 0x00000000000001f0 0x00000000000001f0  RW     0x8
  NOTE           0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000020 0x0000000000000020  R      0x8
  NOTE           0x0000000000000358 0x0000000000000358 0x0000000000000358
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_PROPERTY   0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000020 0x0000000000000020  R      0x8
  GNU_EH_FRAME   0x0000000000002014 0x0000000000002014 0x0000000000002014
                 0x0000000000000044 0x0000000000000044  R      0x4
```



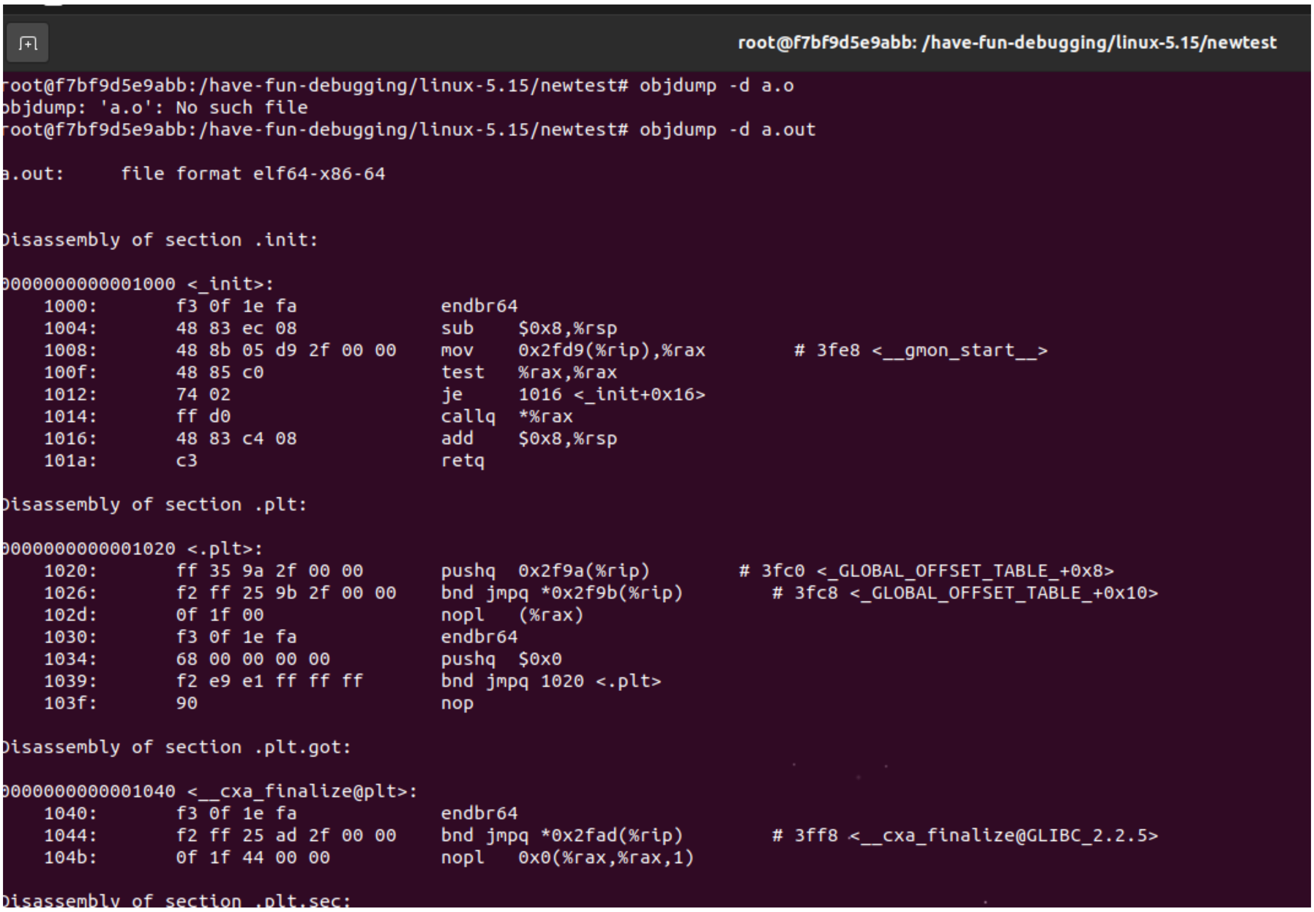
```
   11
   12     .init_array .fini_array .dynamic .got

Dynamic section at offset 0x2dc8 contains 27 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000c (INIT)               0x1000
 0x000000000000000d (FINI)               0x11e8
 0x0000000000000019 (INIT_ARRAY)         0x3db8
 0x000000000000001b (INIT_ARRAYSZ)       8 (bytes)
 0x000000000000001a (FINI_ARRAY)         0x3dc0
 0x000000000000001c (FINI_ARRAYSZ)       8 (bytes)
 0x000000006ffffef5 (GNU_HASH)           0x3a0
 0x0000000000000005 (STRTAB)             0x470
 0x0000000000000006 (SYMTAB)             0x3c8
 0x000000000000000a (STRSZ)              132 (bytes)
 0x000000000000000b (SYMENT)             24 (bytes)
 0x0000000000000015 (DEBUG)              0x0
 0x0000000000000003 (PLTGOT)             0x3fb8
 0x0000000000000002 (PLTRELSZ)           24 (bytes)
 0x0000000000000014 (PLTREL)             RELA
 0x0000000000000017 (JMPREL)             0x5e8
 0x0000000000000007 (RELA)               0x528
 0x0000000000000008 (RELASZ)             192 (bytes)
 0x0000000000000009 (RELAENT)            24 (bytes)
 0x000000000000001e (FLAGS)              BIND_NOW
 0x000000006ffffffb (FLAGS_1)            Flags: NOW PIE
 0x000000006ffffffe (VERNEED)            0x508
 0x000000006fffffff (VERNEEDNUM)         1
 0x000000006ffffff0 (VERSYM)             0x4f4
 0x000000006ffffff9 (RELACOUNT)          3
 0x0000000000000000 (NULL)               0x0

Relocation section '.rela.dyn' at offset 0x528 contains 8 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000003db8  000000000008 R_X86_64_RELATIVE                    1140
000000003dc0  000000000008 R_X86_64_RELATIVE                    1100
000000004008  000000000008 R_X86_64_RELATIVE                    4008
```

（后信息省略）

odjdump

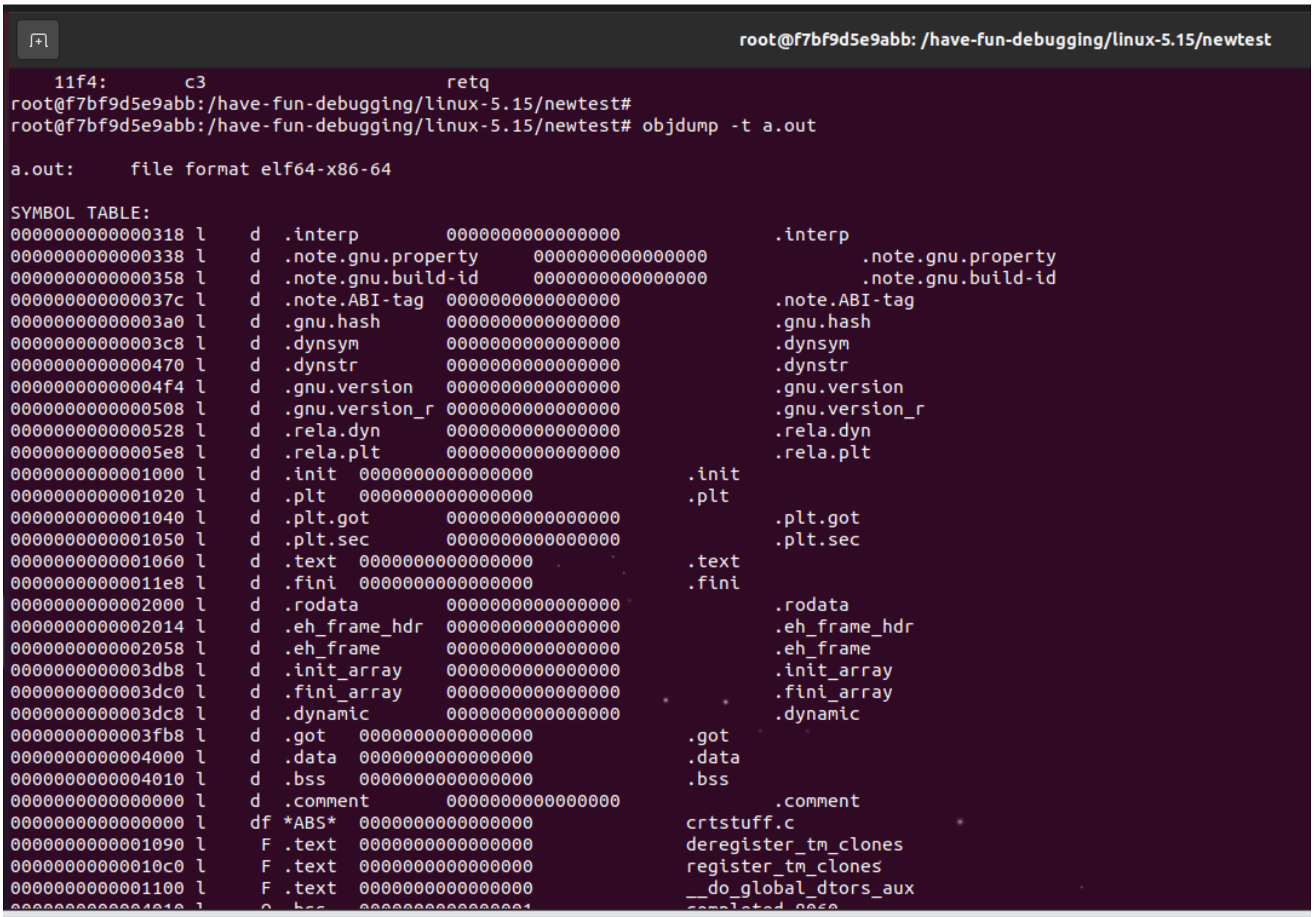
```
objdump -d a.out //-d 参数看代码段反汇编结果
objdump -t a.out //显示文件的符号表入口。
objdump -t libc.a grep -w printf //查找 printf 在 libc.a 库的哪个目标文件
objdump -h simple.o //显示目标文件各个section的头部摘要信息
objdump -r simple.o //查看重定位表
objdump -f simple.o //显示objfile中每个文件的整体头部摘要信息
objdump -s simple.o //显示指定section的完整内容
objdump -x simple.o //显示所可用的头信息
objdump -a simple.o //显示档案库的成员信息
```

反汇编结果如下



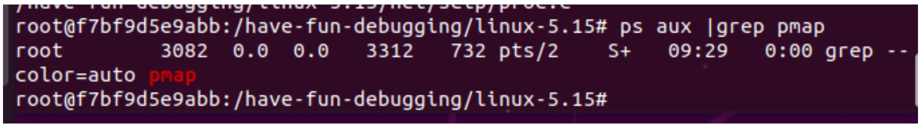
```
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# objdump -d a.o
objdump: 'a.o': No such file
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# objdump -d a.out

a.out:     file format elf64-x86-64


Disassembly of section .init:

0000000000001000 <_init>:
    1000:       f3 0f 1e fa             endbr64
    1004:       48 83 ec 08             sub    $0x8,%rsp
    1008:       48 8b 05 d9 2f 00 00    mov    0x2fd9(%rip),%rax        # 3fe8 <__gmon_start__>
    100f:       48 85 c0                test   %rax,%rax
    1012:       74 02                   je     1016 <_init+0x16>
    1014:       ff d0                   callq  *%rax
    1016:       48 83 c4 08             add    $0x8,%rsp
    101a:       c3                      retq

Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:       ff 35 9a 2f 00 00       pushq  0x2f9a(%rip)        # 3fc0 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:       f2 ff 25 9b 2f 00 00    bnd jmpq *0x2f9b(%rip)        # 3fc8 <_GLOBAL_OFFSET_TABLE_+0x10>
    102d:       0f 1f 00                nopl   (%rax)
    1030:       f3 0f 1e fa             endbr64
    1034:       68 00 00 00 00          pushq  $0x0
    1039:       f2 e9 e1 ff ff ff       bnd jmpq 1020 <.plt>
    103f:       90                      nop

Disassembly of section .plt.got:

0000000000001040 <__cxa_finalize@plt>:
    1040:       f3 0f 1e fa             endbr64
    1044:       f2 ff 25 ad 2f 00 00    bnd jmpq *0x2fad(%rip)        # 3ff8 <__cxa_finalize@GLIBC_2.2.5>
    104b:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:
```

符号表入口



```
    11f4:       c3                      retq
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest#
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# objdump -t a.out

a.out:     file format elf64-x86-64

SYMBOL TABLE:
0000000000000318 l    d  .interp        0000000000000000              .interp
0000000000000338 l    d  .note.gnu.property     0000000000000000              .note.gnu.property
0000000000000358 l    d  .note.gnu.build-id     0000000000000000              .note.gnu.build-id
000000000000037c l    d  .note.ABI-tag  0000000000000000              .note.ABI-tag
00000000000003a0 l    d  .gnu.hash      0000000000000000              .gnu.hash
00000000000003c8 l    d  .dynsym        0000000000000000              .dynsym
0000000000000470 l    d  .dynstr        0000000000000000              .dynstr
00000000000004f4 l    d  .gnu.version   0000000000000000              .gnu.version
0000000000000508 l    d  .gnu.version_r 0000000000000000              .gnu.version_r
0000000000000528 l    d  .rela.dyn      0000000000000000              .rela.dyn
00000000000005e8 l    d  .rela.plt      0000000000000000              .rela.plt
0000000000001000 l    d  .init  0000000000000000              .init
0000000000001020 l    d  .plt   0000000000000000              .plt
0000000000001040 l    d  .plt.got       0000000000000000              .plt.got
0000000000001050 l    d  .plt.sec       0000000000000000              .plt.sec
0000000000001060 l    d  .text  0000000000000000              .text
00000000000011e8 l    d  .fini  0000000000000000              .fini
0000000000002000 l    d  .rodata        0000000000000000              .rodata
0000000000002014 l    d  .eh_frame_hdr  0000000000000000              .eh_frame_hdr
0000000000002058 l    d  .eh_frame      0000000000000000              .eh_frame
0000000000003db8 l    d  .init_array    0000000000000000              .init_array
0000000000003dc0 l    d  .fini_array    0000000000000000              .fini_array
0000000000003dc8 l    d  .dynamic       0000000000000000              .dynamic
0000000000003fb8 l    d  .got   0000000000000000              .got
0000000000004000 l    d  .data  0000000000000000              .data
0000000000004010 l    d  .bss   0000000000000000              .bss
0000000000000000 l    d  .comment       0000000000000000              .comment
0000000000000000 l    df *ABS*  0000000000000000              crtstuff.c
0000000000001090 l     F .text  0000000000000000              deregister_tm_clones
00000000000010c0 l     F .text  0000000000000000              register_tm_clones
0000000000001100 l     F .text  0000000000000000              __do_global_dtors_aux
0000000000004010 l     O .bss   0000000000000001              completed.8060
```

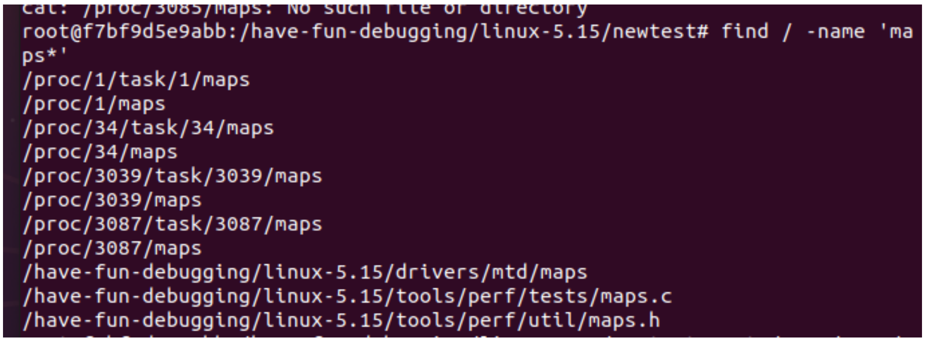
Run an ELF file and cat /proc/PID/maps to give its memory layout.



```
/have-fun-debugging/linux-5.15/net/sctp/proc.c
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15# ps aux |grep pmap
root        3082  0.0  0.0   3312   732 pts/2    S+   09:29   0:00 grep --
color=auto pmap
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15#
```

```
#ps aux |grep pmap //查看程序pid
```

可以看到程序pmap的pid为3082，但是这个程序运行结束后，它所分配的内存也被回收了，所以在下一步搜索中并看不到/3082/maps。此时文件夹中并没有/proc路径，但是经过搜索我们可以看到这个路径是在系统根目录下的。



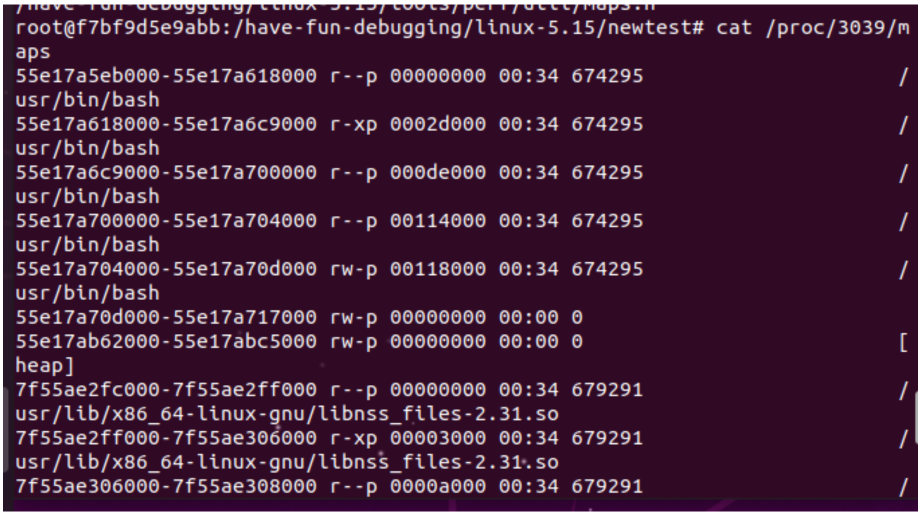
```
cat: /proc/3085/maps: No such file or directory
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# find / -name 'ma
ps*'
/proc/1/task/1/maps
/proc/1/maps
/proc/34/task/34/maps
/proc/34/maps
/proc/3039/task/3039/maps
/proc/3039/maps
/proc/3087/task/3087/maps
/proc/3087/maps
/have-fun-debugging/linux-5.15/drivers/mtd/maps
/have-fun-debugging/linux-5.15/tools/perf/tests/maps.c
/have-fun-debugging/linux-5.15/tools/perf/util/maps.h
```

通过查找可以发现有这几个进程。其中3087为terminal窗口执行每一条指令的进程，每输入一条指令该数都会增加，查询指令对应的进程号是3087，在此次查询结束后是无法通过cat /proc/3087/maps访问到的。故选择进程3039进行查看。

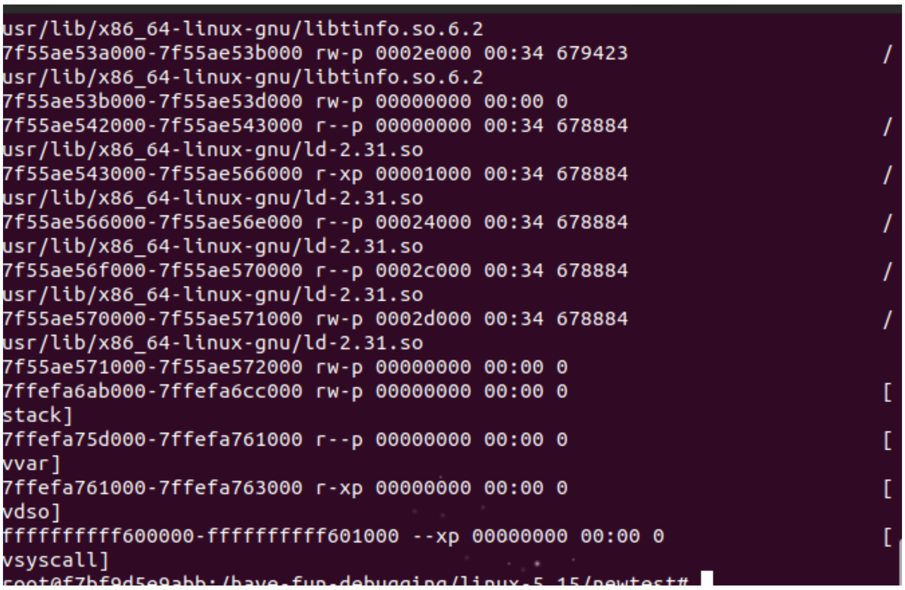
可以看到其中每个进程运行的：

- address: 0085d000-00872000 虚拟内存区域的起始和终止地址文件所占的地址空间
- perms:rw-p 权限：r=read, w=write, x=execute, s=shared, p=private(copy on write)
- offset: 00000000 虚拟内存区域在被映射文件中的偏移量
- dev: 03:08 文件的主设备号和次设备号

- inode: 设备的节点号，0表示没有节点与内存相对应
- name: /lib/ld-2.2.12.so 被映射文件的文件名



```
/have-fun-debugging/linux-5.15/tools/perf/util/maps.h
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest# cat /proc/3039/m
aps
55e17a5eb000-55e17a618000 r--p 00000000 00:34 674295                     /
usr/bin/bash
55e17a618000-55e17a6c9000 r-xp 0002d000 00:34 674295                     /
usr/bin/bash
55e17a6c9000-55e17a700000 r--p 000de000 00:34 674295                     /
usr/bin/bash
55e17a700000-55e17a704000 r--p 00114000 00:34 674295                     /
usr/bin/bash
55e17a704000-55e17a70d000 rw-p 00118000 00:34 674295                     /
usr/bin/bash
55e17a70d000-55e17a717000 rw-p 00000000 00:00 0
55e17ab62000-55e17abc5000 rw-p 00000000 00:00 0                          [
heap]
7f55ae2fc000-7f55ae2ff000 r--p 00000000 00:34 679291                     /
usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f55ae2ff000-7f55ae306000 r-xp 00003000 00:34 679291                     /
usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
7f55ae306000-7f55ae308000 r--p 0000a000 00:34 679291                     /
```



```
usr/lib/x86_64-linux-gnu/libtinfo.so.6.2
7f55ae53a000-7f55ae53b000 rw-p 0002e000 00:34 679423                     /
usr/lib/x86_64-linux-gnu/libtinfo.so.6.2
7f55ae53b000-7f55ae53d000 rw-p 00000000 00:00 0
7f55ae542000-7f55ae543000 r--p 00000000 00:34 678884                     /
usr/lib/x86_64-linux-gnu/ld-2.31.so
7f55ae543000-7f55ae566000 r-xp 00001000 00:34 678884                     /
usr/lib/x86_64-linux-gnu/ld-2.31.so
7f55ae566000-7f55ae56e000 r--p 00024000 00:34 678884                     /
usr/lib/x86_64-linux-gnu/ld-2.31.so
7f55ae56f000-7f55ae570000 r--p 0002c000 00:34 678884                     /
usr/lib/x86_64-linux-gnu/ld-2.31.so
7f55ae570000-7f55ae571000 rw-p 0002d000 00:34 678884                     /
usr/lib/x86_64-linux-gnu/ld-2.31.so
7f55ae571000-7f55ae572000 rw-p 00000000 00:00 0
7ffefa6ab000-7ffefa6cc000 rw-p 00000000 00:00 0                          [
stack]
7ffefa75d000-7ffefa761000 r--p 00000000 00:00 0                          [
vvar]
7ffefa761000-7ffefa763000 r-xp 00000000 00:00 0                          [
vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [
vsyscall]
root@f7bf9d5e9abb:/have-fun-debugging/linux-5.15/newtest#
```

2, 3, 4 need to have screenshots.