Lab 1: RV64 内核引导

学号：3200105088　　姓名 巩德志

1 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。

2 实验环境

- Environment in Lab0

3 实验步骤

3.1 准备工程

学习riscv汇编、makefile、内联汇编等知识，完善以下文件：

- arch/riscv/kernel/head.S
- lib/Makefile
- arch/riscv/kernel/sbi.c
- lib/print.c
- arch/riscv/include/defs.h

需要实现调用sbi_ecall，完成字符串输出puts()和puti()的实现。

3.2 编写head.S

```
.extern start_kernel

    .section .text.entry
    .globl _start
_start:
    # ------------------
    # - your code here -
    la sp, boot_stack_top
    j start_kernel
    # ------------------

    .section .bss.stack
    .globl boot_stack
boot_stack:
    .space 0x1000 # <-- change to your stack size

    .globl boot_stack_top
boot_stack_top:
```

首先`la sp, boot_stack_top`将boot_stack_top载入保存栈指针的寄存器sp，实现了将该栈放置在`.bss.stack`段。`j start_kernel`通过跳转指令跳转到`start_kernel`函数

3.3 完善 Makefile 脚本

```makefile
all : print.o

print.o : print.c
    $(GCC) $(CFLAG) -o print.o -c print.c

clean :
    rm print.o
```

链接print.c生成print.o文件

3.4 补充 `sbi.c`

```c
#include "types.h"
#include "sbi.h"


struct sbiret sbi_ecall(int ext, int fid, uint64 arg0,
                        uint64 arg1, uint64 arg2,
                        uint64 arg3, uint64 arg4,
                        uint64 arg5)
{
    // unimplemented
    struct sbiret res;

    __asm__ volatile(
        "mv a0, %[arg0]\n"
        "mv a1, %[arg1]\n"
        "mv a2, %[arg2]\n"
        "mv a3, %[arg3]\n"
        "mv a4, %[arg4]\n"
        "mv a5, %[arg5]\n"
        "mv a6, %[fid]\n"
        "mv a7, %[ext]\n"
        "ecall\n"
        "mv %[error], a0\n"
        "mv %[value], a1\n"
        :[error] "=r" (res.error), [value] "=r" (res.value)
        :[arg0] "r" (arg0), [arg1] "r" (arg1), [arg2] "r" (arg2), [arg3] "r" (arg3), [arg4] "r" (arg4), [arg5] "r" (arg5), [ext] "r" (ext), [fid] "r" (fid)
        : "memory", "a0", "a1", "a2", "a3", "a4", "a5", "a6", "a7"
    );
    return res;
}
```

```
        "mv a0, %[arg0]\n"
        "mv a1, %[arg1]\n"
        "mv a2, %[arg2]\n"
        "mv a3, %[arg3]\n"
        "mv a4, %[arg4]\n"
        "mv a5, %[arg5]\n"
        "mv a6, %[fid]\n"
        "mv a7, %[ext]\n"
```

将参数arg0-arg7分别存入a0-a5寄存器，将fid ext存入a6 a7

```
        "ecall\n"
```

调用ecall函数，将七个参数传给ecall，进行字符串输出操作

```
    "mv %[error], a0\n"
    "mv %[value], a1\n"
```

ecall执行后的返回值保存在a0 a1，将返回值存入error和value

```
        :[error] "=r" (res.error), [value] "=r" (res.value)
```

输出结果存入res.error、res.value

```
        :[arg0] "r" (arg0), [arg1] "r" (arg1), [arg2] "r" (arg2), [arg3] "r"
(arg3), [arg4] "r" (arg4), [arg5] "r" (arg5), [ext] "r" (ext), [fid] "r" (fid)
```

输入arg0-arg5、ext、fid等分别放入临时变量[arg0],[ext],[fid]

```
: "memory", "a0", "a1", "a2", "a3", "a4", "a5", "a6", "a7"
```

这句表示进行过输出输出的寄存器和内存有以上几个

3.5 `puts()` 和 `puti()`

```
#include "print.h"
#include "sbi.h"

void puts(char *s) {
    // unimplemented
    while(*s)
    {
        sbi_ecall(0x1, 0x0, *s, 0, 0, 0, 0, 0);
        s++;
    }
}

void puti(int x) {
    // unimplemented
    int bit[100];
    int count = 0;
    while(x)
    {
        bit[count++] = x % 10;
        x /= 10;
    }
    for(int i = count - 1; i >= 0; i--)
        sbi_ecall(0x1, 0x0, bit[i]+'0', 0, 0, 0, 0, 0);
}
```

调用sbi_ecall打印字符，前三个参数分别代表ExtensionID，FunctionID，ascii码

3.6 修改 defs

```
#ifndef _DEFS_H
#define _DEFS_H

#include "types.h"

#define csr_read(csr)                       \
({                                          \
    register uint64 __v;                    \
    /* unimplemented */                     \
    asm volatile ("csrr %0, " #csr           \
                    : "=r"(__v)             \
                    :                       \
                    : "memory"   );         \
    __v;                                    \
})

#define csr_write(csr, val)                         \
({                                                  \
    uint64 __v = (uint64)(val);                     \
    asm volatile ("csrw " #csr ", %0"               \
                    : : "r" (__v)                   \
                    : "memory");                    \
})

#endif
```

4 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?

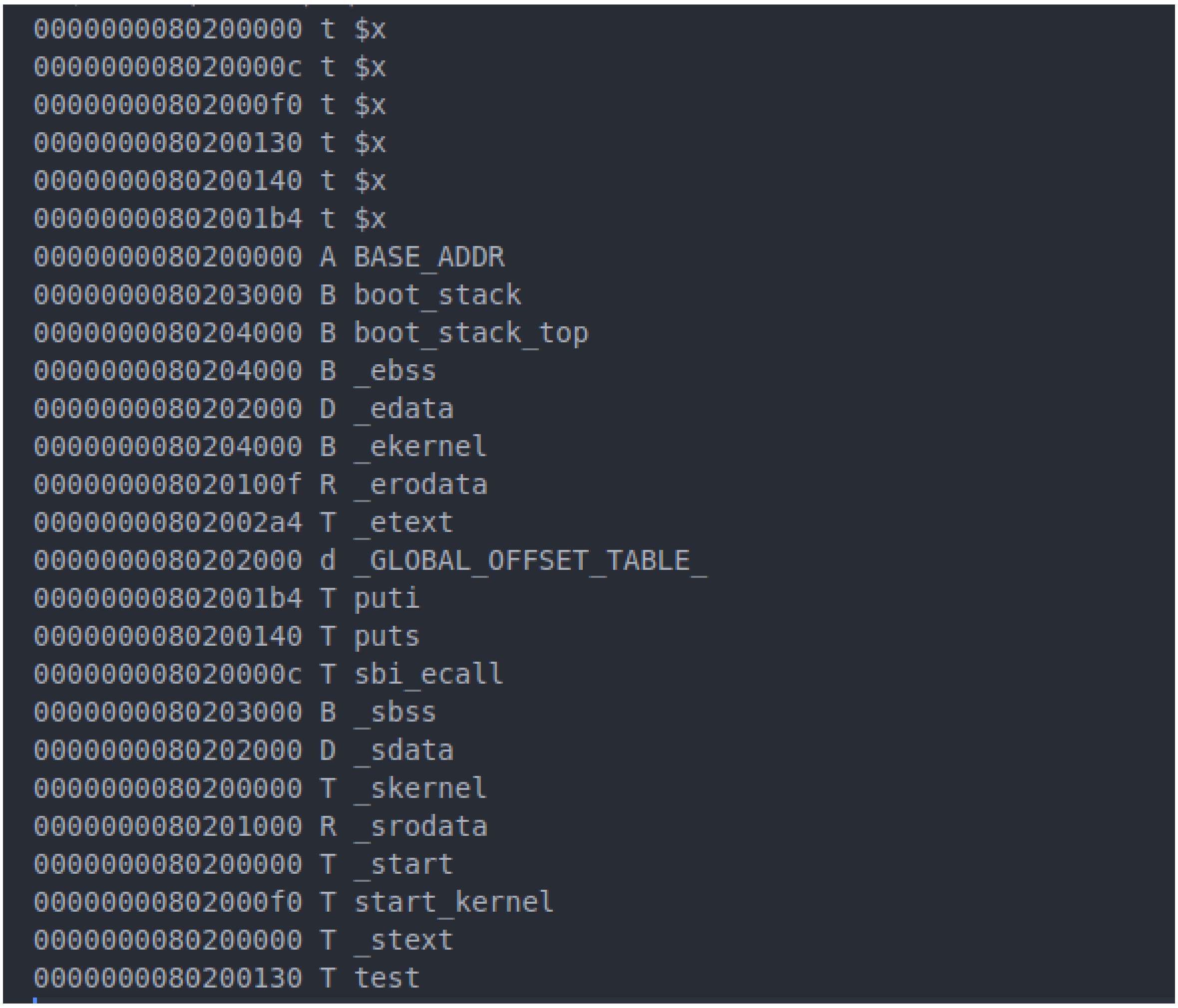
RISC-V 的 calling convention：

1. 如果函数的参数是结构体成员，那么每个参数要按照所在平台上指针类型大小对齐。结构体至多8个成员会被放在寄存器中，剩余的部分被存放在栈上，sp指向第一个没有被存放在寄存器上的结构体成员。
2. 小于1个指针字长的参数被存在寄存器的低位，如果存在栈上也是存放在内存的低地址上。
3. 函数返回值如果是基本类型或者只包含一两个成员的基本结构体的成员，存放在相应的整形寄存器a0和a1，浮点寄存器fa0和fa1. 大于两个指针字长的返回值存放在内存上。
4. 栈是从高地址向低地址方向的， 并且是16字节对齐的。
5. 寄存器t0$_{\text{t6，ft0}}$ft11被称为临时寄存器，由调用者保存。s0~s11, fs0~fs11由被调者保存。

Callee-saved register用于保存应在每次调用中保留的长寿命值。当调用者进行过程调用时，可以期望这些寄存器在被调用者返回后将保持相同的值，这使被调用者有责任在返回调用者之前保存它们并恢复它们。

Caller-saved register用于保存不需要在各个调用之间保留的临时数据。因此，如果要在过程调用后恢复该值，则调用方有责任将这些寄存器压入堆栈或将其复制到其他位置。不过，让调用销毁这些寄存器中的临时值是正常的。从被调用方的角度来看，函数可以自由覆盖（也就是破坏）这些寄存器，而无需保存/恢复。

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



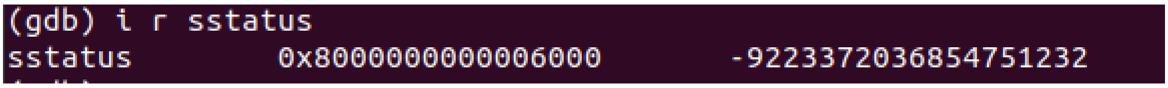
```
0000000080200000 t $x
000000008020000c t $x
00000000802000f0 t $x
0000000080200130 t $x
0000000080200140 t $x
00000000802001b4 t $x
0000000080200000 A BASE_ADDR
0000000080203000 B boot_stack
0000000080204000 B boot_stack_top
0000000080204000 B _ebss
0000000080202000 D _edata
0000000080204000 B _ekernel
000000008020100f R _erodata
00000000802002a4 T _etext
0000000080202000 d _GLOBAL_OFFSET_TABLE_
00000000802001b4 T puti
0000000080200140 T puts
000000008020000c T sbi_ecall
0000000080203000 B _sbss
0000000080202000 D _sdata
0000000080200000 T _skernel
0000000080201000 R _srodata
0000000080200000 T _start
00000000802000f0 T start_kernel
0000000080200000 T _stext
0000000080200130 T test
```

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）。

调用宏读取sstatus的值

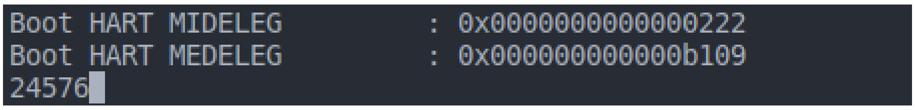
```
#include "print.h"
#include "sbi.h"
#include "defs.h"

extern void test();

int start_kernel() {
    uint64 res = csr_read(sstatus);
    puti(res);
    csr_write(sscratch, 0x0100);

    test(); // DO NOT DELETE !!!

    return 0;
}
```

查看寄存器的值



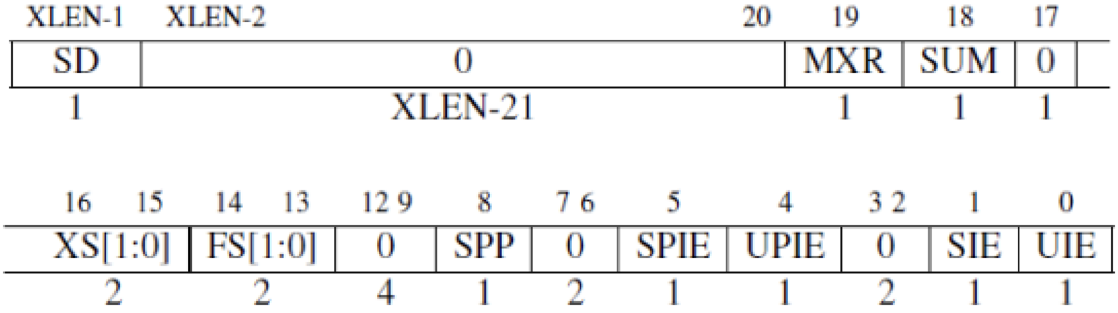
```
(gdb) i r sstatus
sstatus        0x8000000000006000       -9223372036854751232
```

查看运行结果（读取的值）



```
Boot HART MIDELEG          : 0x0000000000000222
Boot HART MEDELEG          : 0x000000000000b109
24576
```

sstatus是一个SXLEN-bit 读写寄存器，状态寄存器用来存放两类信息：一类是体现当前指令执行结果的各种状态信息，另一类是存放控制信息



XLEN-1	XLEN-2 20	19	18	17
SD	0	MXR	SUM	0
1	XLEN-21	1	1	1

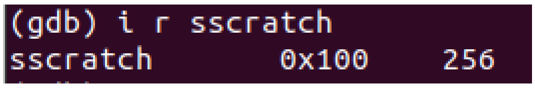
16 15	14 13	12 9	8	7 6	5	4	3 2	1	0
XS[1:0]	FS[1:0]	0	SPP	0	SPIE	UPIE	0	SIE	UIE
2	2	4	1	2	1	1	2	1	1

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）。

```
csr_write(sscratch, 0x0100);
```

向sscratch写入256

查看sscratch的值



```
(gdb) i r sscratch
sscratch       0x100    256
```

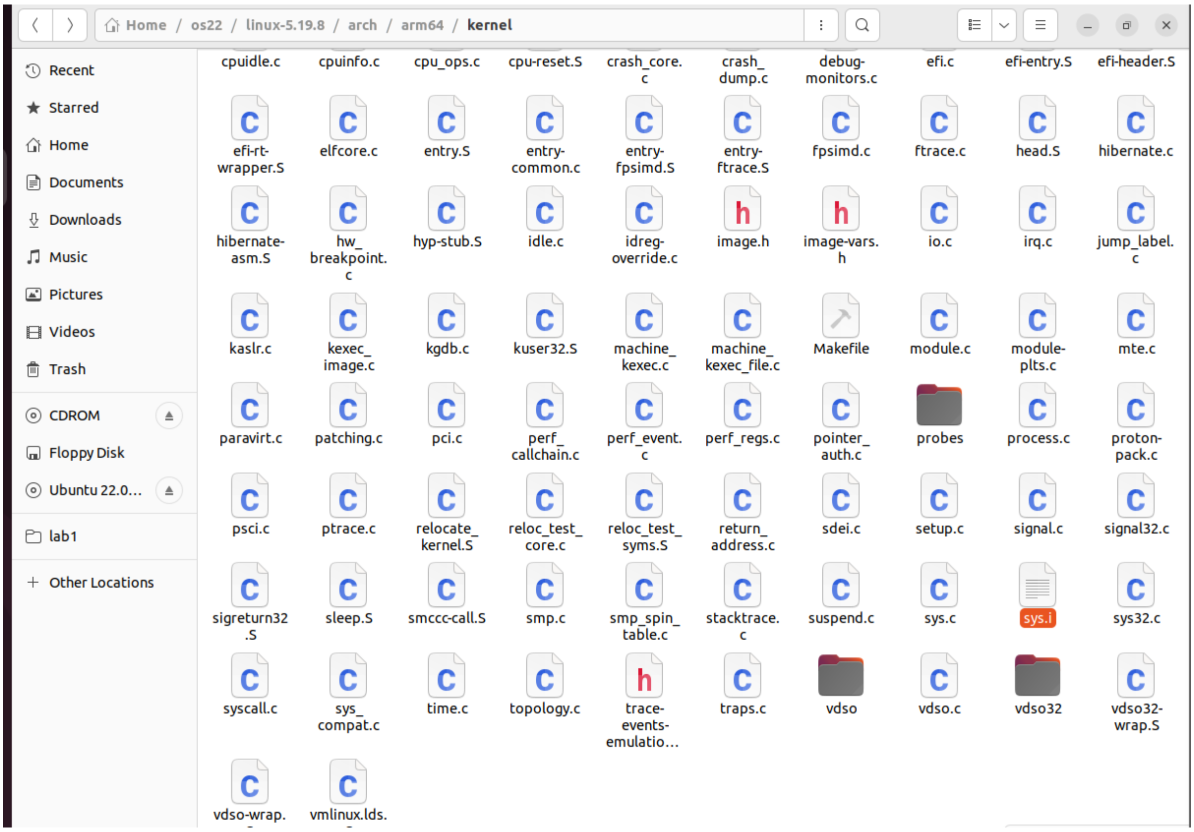
5 Detail your steps about how to get `arch/arm64/kernel/sys.i`

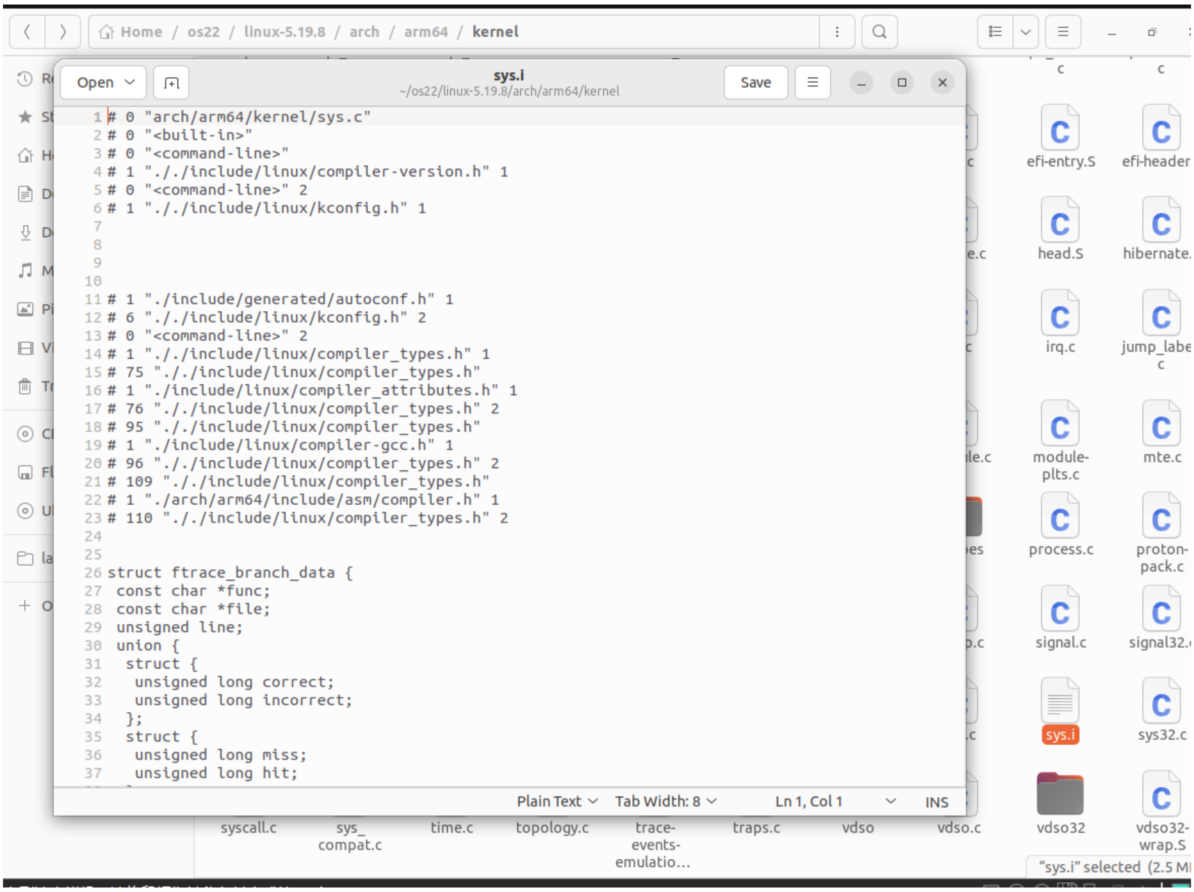
安装交叉编译工具链

```
sudo apt install gcc-aarch64-linux-gnu
```

获得编译预处理产物

```
# 先 config
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- defconfig

# 然后指定要生成的文件
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- ./arch/arm64/kernel/sys.i
```



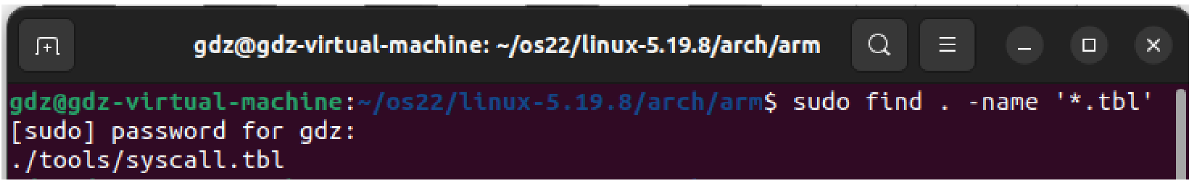

Home / os22 / linux-5.19.8 / arch / arm64 / kernel
Open
Save
sys.i
~/os22/linux-5.19.8/arch/arm64/kernel
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
"sys.i" selected (2.5 MB

```
# 0 "arch/arm64/kernel/sys.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "./include/linux/compiler-version.h" 1
# 0 "<command-line>" 2
# 1 "./include/linux/kconfig.h" 1




# 1 "./include/generated/autoconf.h" 1
# 6 "./include/linux/kconfig.h" 2
# 0 "<command-line>" 2
# 1 "./include/linux/compiler_types.h" 1
# 75 "./include/linux/compiler_types.h"
# 1 "./include/linux/compiler_attributes.h" 1
# 76 "./include/linux/compiler_types.h" 2
# 95 "./include/linux/compiler_types.h"
# 1 "./include/linux/compiler-gcc.h" 1
# 96 "./include/linux/compiler_types.h" 2
# 109 "./include/linux/compiler_types.h"
# 1 "./arch/arm64/include/asm/compiler.h" 1
# 110 "./include/linux/compiler_types.h" 2


struct ftrace_branch_data {
 const char *func;
 const char *file;
 unsigned line;
 union {
  struct {
   unsigned long correct;
   unsigned long incorrect;
  };
  struct {
   unsigned long miss;
   unsigned long hit;
```

6 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.

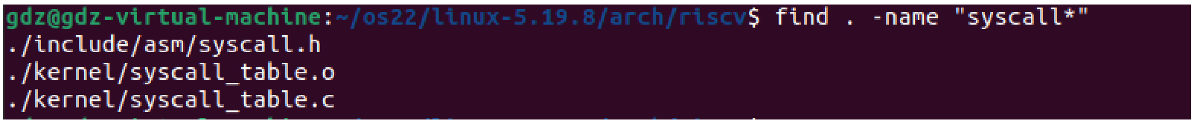
```
sudo find . -name '*.tbl'
sudo find . -name "syscall*"
```

ARM32:



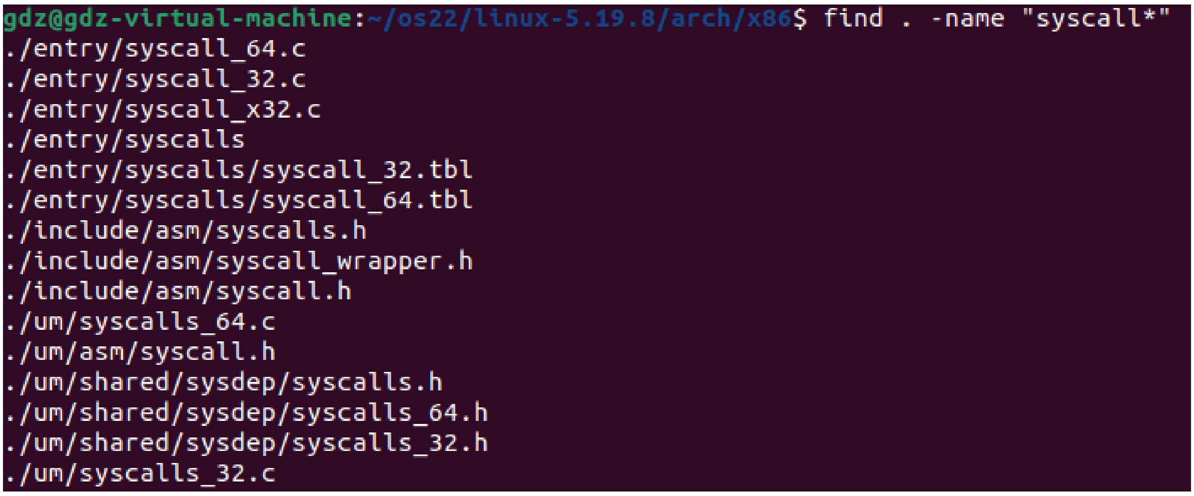
```
gdz@gdz-virtual-machine:~/os22/linux-5.19.8/arch/arm$ sudo find . -name '*.tbl'
[sudo] password for gdz:
./tools/syscall.tbl
```

risc-v(32/64):(不含tbl文件)



```
gdz@gdz-virtual-machine:~/os22/linux-5.19.8/arch/riscv$ find . -name "syscall*"
./include/asm/syscall.h
./kernel/syscall_table.o
./kernel/syscall_table.c
```

x86(32/64):



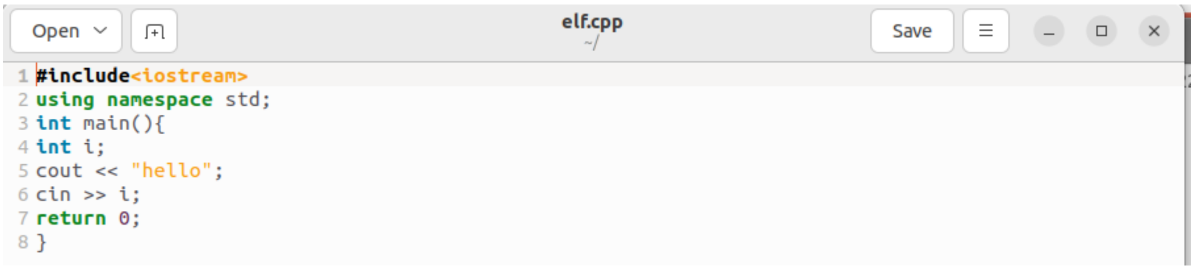
```
gdz@gdz-virtual-machine:~/os22/linux-5.19.8/arch/x86$ find . -name "syscall*"
./entry/syscall_64.c
./entry/syscall_32.c
./entry/syscall_x32.c
./entry/syscalls
./entry/syscalls/syscall_32.tbl
./entry/syscalls/syscall_64.tbl
./include/asm/syscalls.h
./include/asm/syscall_wrapper.h
./include/asm/syscall.h
./um/syscalls_64.c
./um/asm/syscall.h
./um/shared/sysdep/syscalls.h
./um/shared/sysdep/syscalls_64.h
./um/shared/sysdep/syscalls_32.h
./um/syscalls_32.c
```

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 是一类文件类型，而不是特指某一后缀的文件。 ELF 文件格式在 Linux 下主要有如下三种文件：可执行文件（.out）、可重定位文件、（.o文件）共享目标文件（.so）

ELF文件由4部分组成，分别是ELF头、程序头表、节和节头表。

编写一个cpp程序



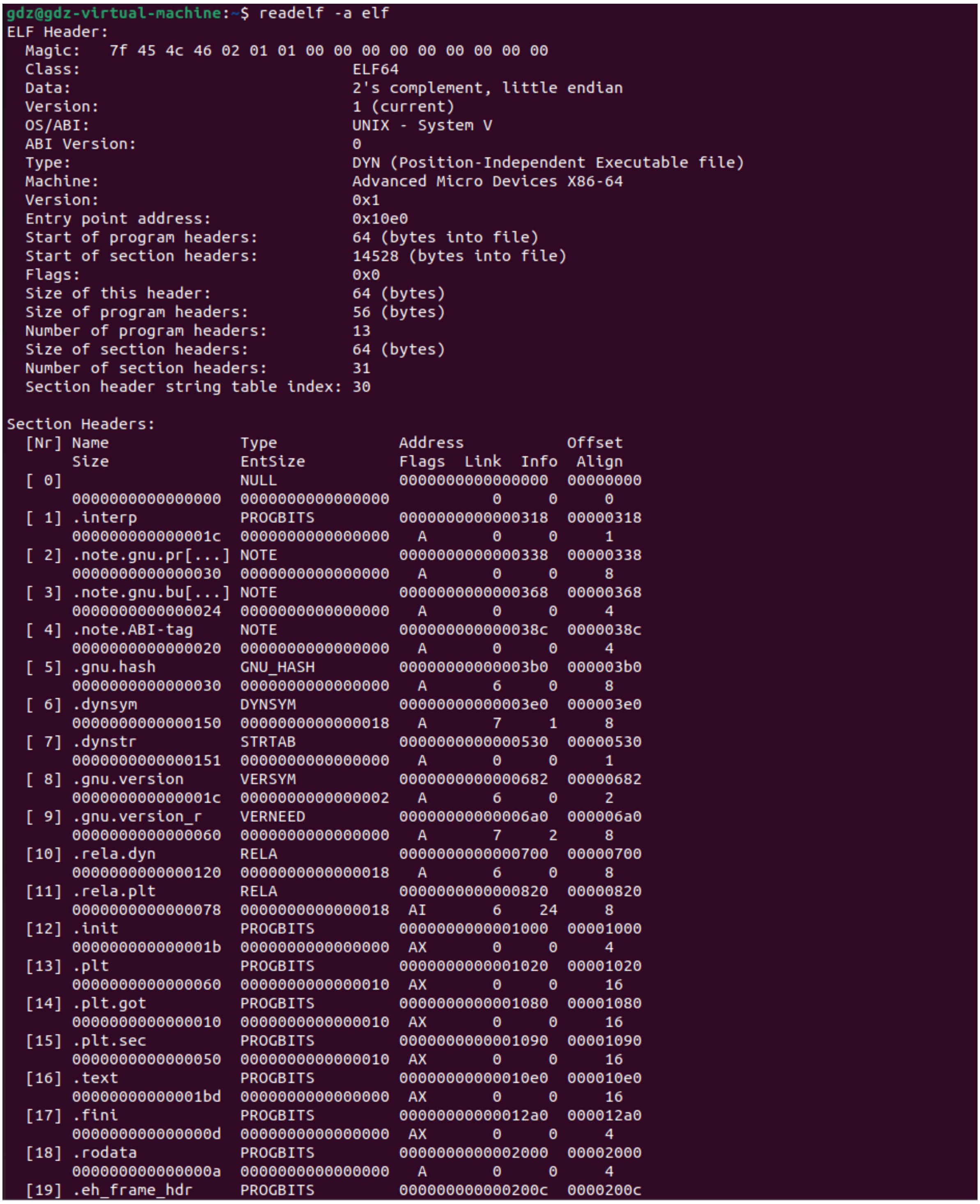
```cpp
#include<iostream>
using namespace std;
int main(){
int i;
cout << "hello";
cin >> i;
return 0;
}
```

使用g++进行编译

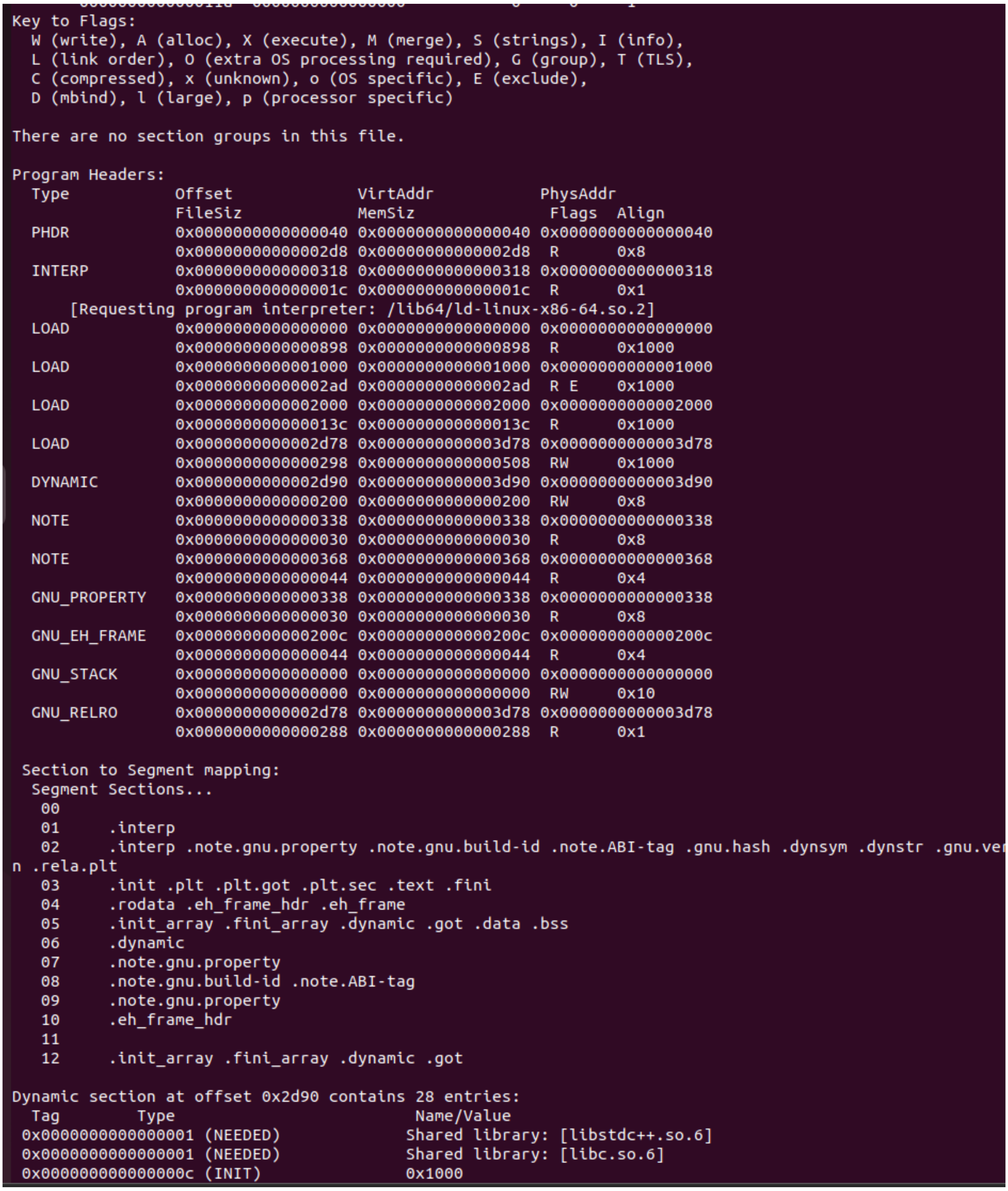
`g++ -o elf elf.cpp`

使用readelf读取hello可执行文件

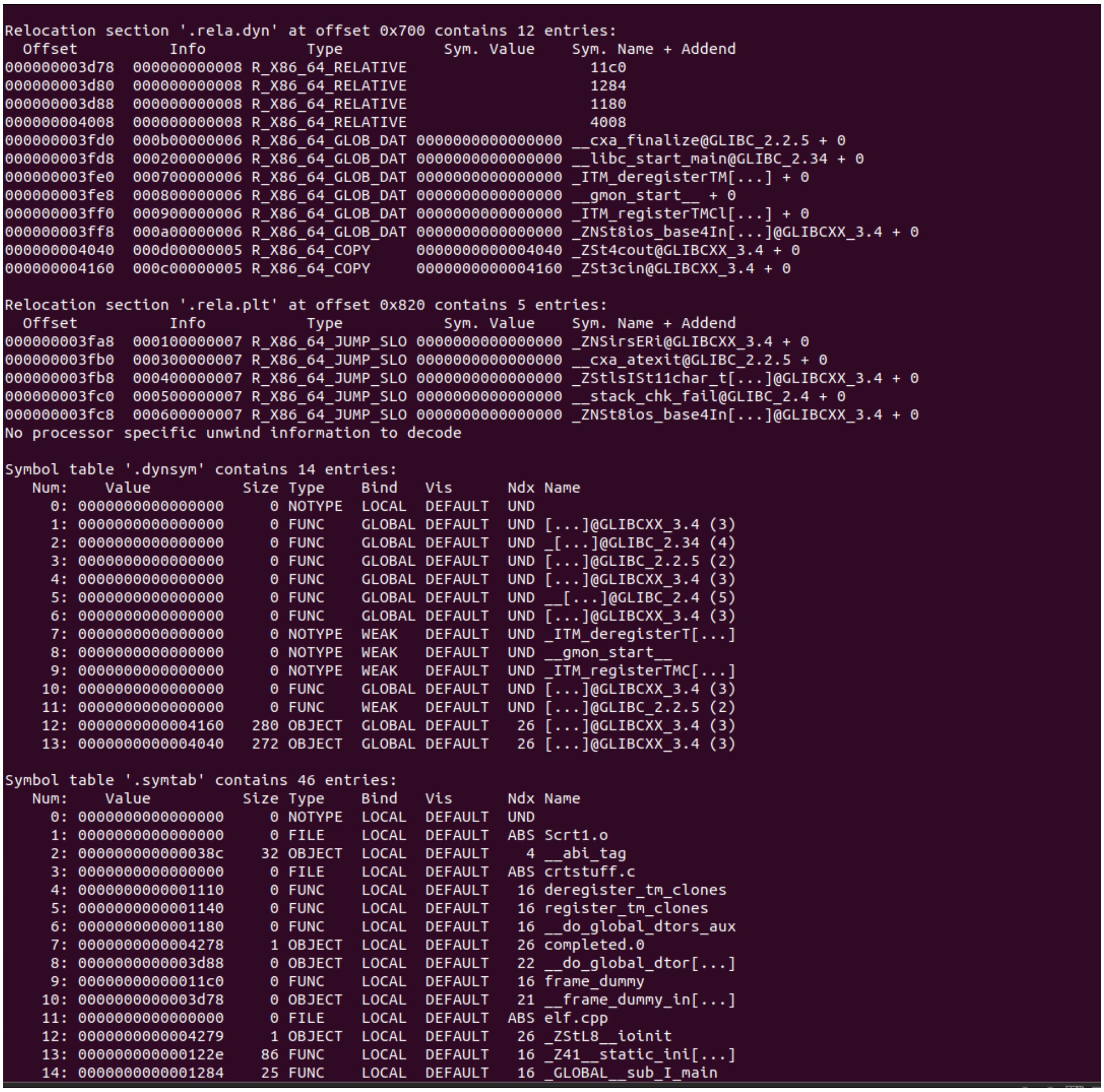
readelf -a elf



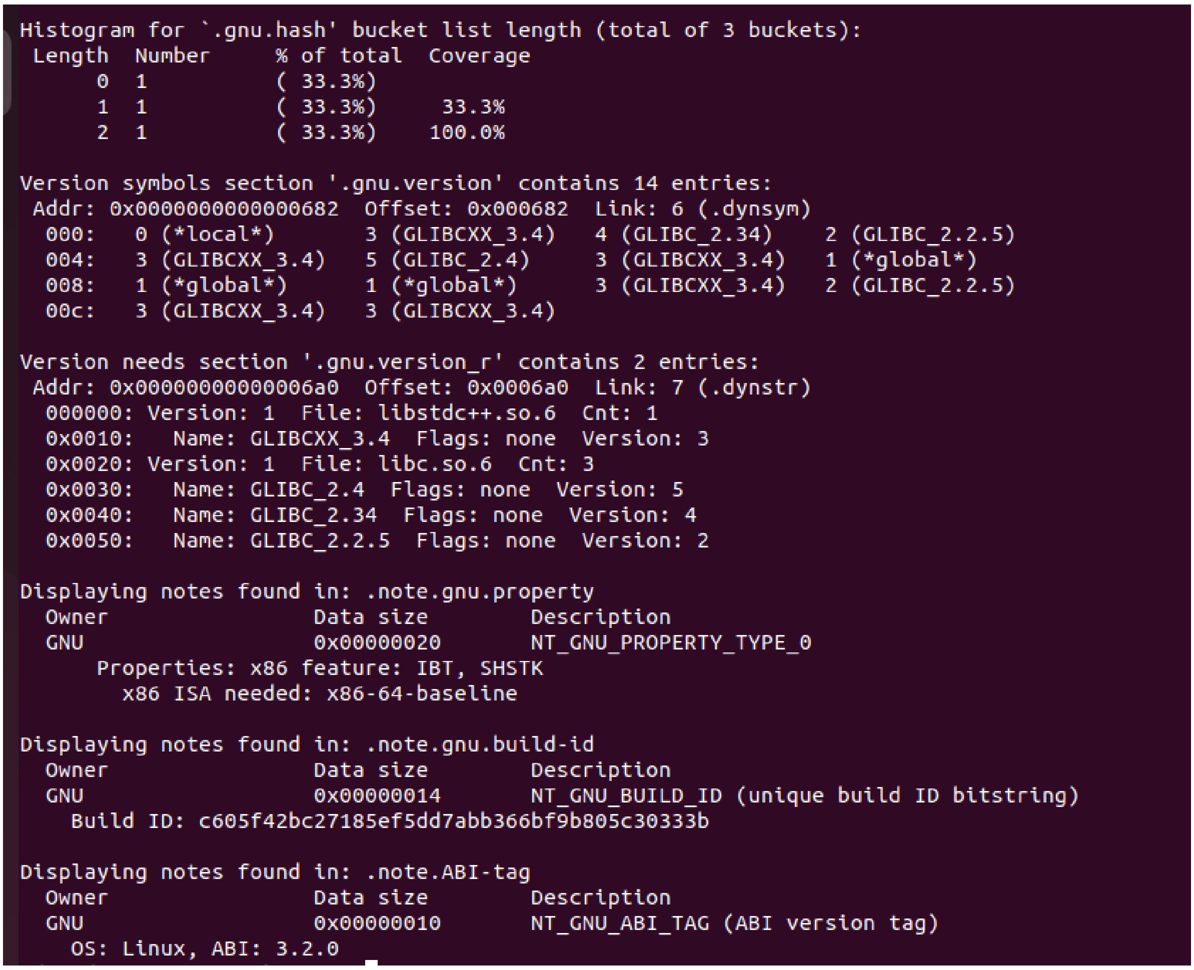
```
gdz@gdz-virtual-machine:~$ readelf -a elf
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x10e0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14528 (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.pr[...] NOTE             0000000000000338  00000338
       0000000000000030  0000000000000000   A       0     0     8
  [ 3] .note.gnu.bu[...] NOTE             0000000000000368  00000368
       0000000000000024  0000000000000000   A       0     0     4
  [ 4] .note.ABI-tag     NOTE             000000000000038c  0000038c
       0000000000000020  0000000000000000   A       0     0     4
  [ 5] .gnu.hash         GNU_HASH         00000000000003b0  000003b0
       0000000000000030  0000000000000000   A       6     0     8
  [ 6] .dynsym           DYNSYM           00000000000003e0  000003e0
       0000000000000150  0000000000000018   A       7     1     8
  [ 7] .dynstr           STRTAB           0000000000000530  00000530
       0000000000000151  0000000000000000   A       0     0     1
  [ 8] .gnu.version      VERSYM           0000000000000682  00000682
       000000000000001c  0000000000000002   A       6     0     2
  [ 9] .gnu.version_r    VERNEED          00000000000006a0  000006a0
       0000000000000060  0000000000000000   A       7     2     8
  [10] .rela.dyn         RELA             0000000000000700  00000700
       0000000000000120  0000000000000018   A       6     0     8
  [11] .rela.plt         RELA             0000000000000820  00000820
       0000000000000078  0000000000000018  AI       6    24     8
  [12] .init             PROGBITS         0000000000001000  00001000
       000000000000001b  0000000000000000  AX       0     0     4
  [13] .plt              PROGBITS         0000000000001020  00001020
       0000000000000060  0000000000000010  AX       0     0     16
  [14] .plt.got          PROGBITS         0000000000001080  00001080
       0000000000000010  0000000000000010  AX       0     0     16
  [15] .plt.sec          PROGBITS         0000000000001090  00001090
       0000000000000050  0000000000000010  AX       0     0     16
  [16] .text             PROGBITS         00000000000010e0  000010e0
       00000000000001bd  0000000000000000  AX       0     0     16
  [17] .fini             PROGBITS         00000000000012a0  000012a0
       000000000000000d  0000000000000000  AX       0     0     4
  [18] .rodata           PROGBITS         0000000000002000  00002000
       000000000000000a  0000000000000000   A       0     0     4
  [19] .eh_frame_hdr     PROGBITS         000000000000200c  0000200c
```



```
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),
  D (mbind), 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
                 0x0000000000000898 0x0000000000000898  R      0x1000
  LOAD           0x0000000000001000 0x0000000000001000 0x0000000000001000
                 0x00000000000002ad 0x00000000000002ad  R E    0x1000
  LOAD           0x0000000000002000 0x0000000000002000 0x0000000000002000
                 0x000000000000013c 0x000000000000013c  R      0x1000
  LOAD           0x0000000000002d78 0x0000000000003d78 0x0000000000003d78
                 0x0000000000000298 0x0000000000000508  RW     0x1000
  DYNAMIC        0x0000000000002d90 0x0000000000003d90 0x0000000000003d90
                 0x0000000000000200 0x0000000000000200  RW     0x8
  NOTE           0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000030 0x0000000000000030  R      0x8
  NOTE           0x0000000000000368 0x0000000000000368 0x0000000000000368
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_PROPERTY   0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000030 0x0000000000000030  R      0x8
  GNU_EH_FRAME   0x000000000000200c 0x000000000000200c 0x000000000000200c
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
  GNU_RELRO      0x0000000000002d78 0x0000000000003d78 0x0000000000003d78
                 0x0000000000000288 0x0000000000000288  R      0x1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.gnu.property .note.gnu.build-id .note.ABI-tag .gnu.hash .dynsym .dynstr .gnu.ver
n .rela.plt
   03     .init .plt .plt.got .plt.sec .text .fini
   04     .rodata .eh_frame_hdr .eh_frame
   05     .init_array .fini_array .dynamic .got .data .bss
   06     .dynamic
   07     .note.gnu.property
   08     .note.gnu.build-id .note.ABI-tag
   09     .note.gnu.property
   10     .eh_frame_hdr
   11
   12     .init_array .fini_array .dynamic .got

Dynamic section at offset 0x2d90 contains 28 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000c (INIT)               0x1000
```



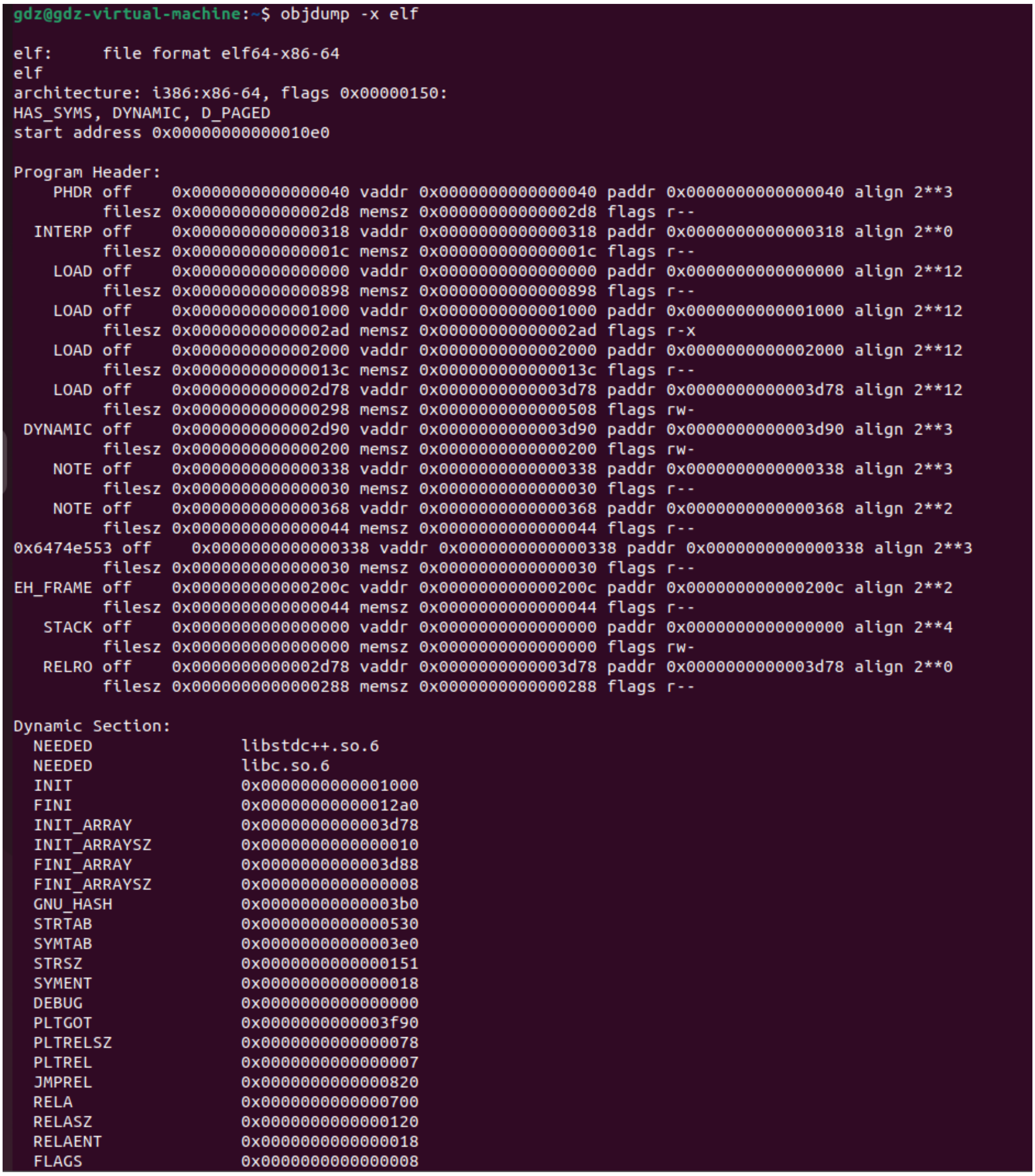
```
Relocation section '.rela.dyn' at offset 0x700 contains 12 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000003d78  000000000008 R_X86_64_RELATIVE                    11c0
000000003d80  000000000008 R_X86_64_RELATIVE                    1284
000000003d88  000000000008 R_X86_64_RELATIVE                    1180
000000004008  000000000008 R_X86_64_RELATIVE                    4008
000000003fd0  000b00000006 R_X86_64_GLOB_DAT 0000000000000000 __cxa_finalize@GLIBC_2.2.5 + 0
000000003fd8  000200000006 R_X86_64_GLOB_DAT 0000000000000000 __libc_start_main@GLIBC_2.34 + 0
000000003fe0  000700000006 R_X86_64_GLOB_DAT 0000000000000000 _ITM_deregisterTM[...] + 0
000000003fe8  000800000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000003ff0  000900000006 R_X86_64_GLOB_DAT 0000000000000000 _ITM_registerTMCl[...] + 0
000000003ff8  000a00000006 R_X86_64_GLOB_DAT 0000000000000000 _ZNSt8ios_base4In[...]@GLIBCXX_3.4 + 0
000000004040  000d00000005 R_X86_64_COPY     0000000000004040 _ZSt4cout@GLIBCXX_3.4 + 0
000000004160  000c00000005 R_X86_64_COPY     0000000000004160 _ZSt3cin@GLIBCXX_3.4 + 0

Relocation section '.rela.plt' at offset 0x820 contains 5 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000003fa8  000100000007 R_X86_64_JUMP_SLO 0000000000000000 _ZNSirsERi@GLIBCXX_3.4 + 0
000000003fb0  000300000007 R_X86_64_JUMP_SLO 0000000000000000 __cxa_atexit@GLIBC_2.2.5 + 0
000000003fb8  000400000007 R_X86_64_JUMP_SLO 0000000000000000 _ZStlsISt11char_t[...]@GLIBCXX_3.4 + 0
000000003fc0  000500000007 R_X86_64_JUMP_SLO 0000000000000000 __stack_chk_fail@GLIBC_2.4 + 0
000000003fc8  000600000007 R_X86_64_JUMP_SLO 0000000000000000 _ZNSt8ios_base4In[...]@GLIBCXX_3.4 + 0
No processor specific unwind information to decode

Symbol table '.dynsym' contains 14 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND 
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBCXX_3.4 (3)
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _[...]@GLIBC_2.34 (4)
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBC_2.2.5 (2)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBCXX_3.4 (3)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __[...]@GLIBC_2.4 (5)
     6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBCXX_3.4 (3)
     7: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
     8: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     9: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMC[...]
    10: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND [...]@GLIBCXX_3.4 (3)
    11: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND [...]@GLIBC_2.2.5 (2)
    12: 0000000000004160   280 OBJECT  GLOBAL DEFAULT   26 [...]@GLIBCXX_3.4 (3)
    13: 0000000000004040   272 OBJECT  GLOBAL DEFAULT   26 [...]@GLIBCXX_3.4 (3)

Symbol table '.symtab' contains 46 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND 
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS Scrt1.o
     2: 000000000000038c    32 OBJECT  LOCAL  DEFAULT    4 __abi_tag
     3: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
     4: 0000000000001110     0 FUNC    LOCAL  DEFAULT   16 deregister_tm_clones
     5: 0000000000001140     0 FUNC    LOCAL  DEFAULT   16 register_tm_clones
     6: 0000000000001180     0 FUNC    LOCAL  DEFAULT   16 __do_global_dtors_aux
     7: 0000000000004278     1 OBJECT  LOCAL  DEFAULT   26 completed.0
     8: 0000000000003d88     0 OBJECT  LOCAL  DEFAULT   22 __do_global_dtor[...]
     9: 00000000000011c0     0 FUNC    LOCAL  DEFAULT   16 frame_dummy
    10: 0000000000003d78     0 OBJECT  LOCAL  DEFAULT   21 __frame_dummy_in[...]
    11: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS elf.cpp
    12: 0000000000004279     1 OBJECT  LOCAL  DEFAULT   26 _ZStL8__ioinit
    13: 000000000000122e    86 FUNC    LOCAL  DEFAULT   16 _Z41__static_ini[...]
    14: 0000000000001284    25 FUNC    LOCAL  DEFAULT   16 _GLOBAL__sub_I_main
```



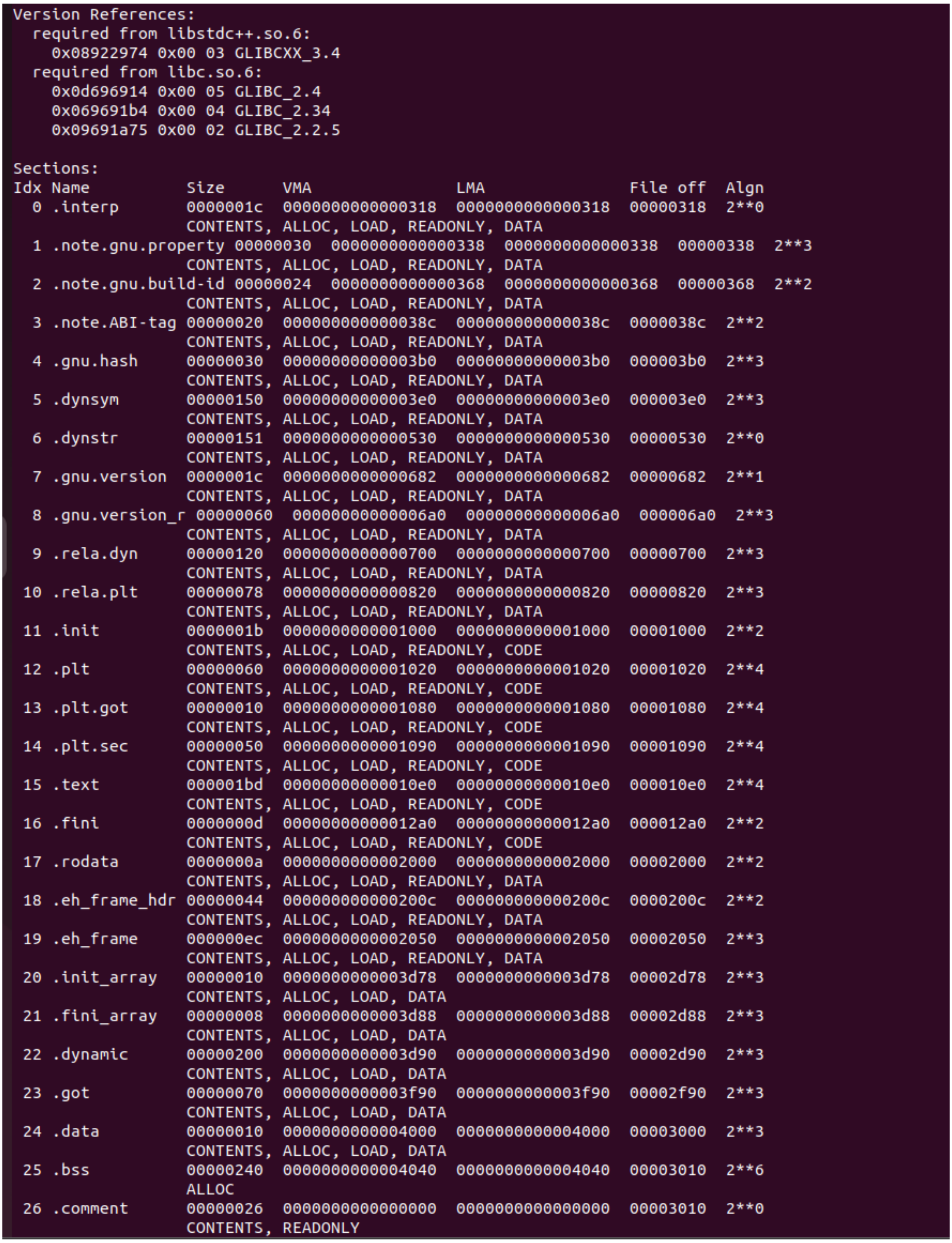
```
Histogram for `.gnu.hash' bucket list length (total of 3 buckets):
 Length  Number     % of total  Coverage
      0  1          ( 33.3%)
      1  1          ( 33.3%)     33.3%
      2  1          ( 33.3%)    100.0%

Version symbols section '.gnu.version' contains 14 entries:
 Addr: 0x0000000000000682  Offset: 0x000682  Link: 6 (.dynsym)
  000:   0 (*local*)       3 (GLIBCXX_3.4)    4 (GLIBC_2.34)    2 (GLIBC_2.2.5)
  004:   3 (GLIBCXX_3.4)   5 (GLIBC_2.4)      3 (GLIBCXX_3.4)   1 (*global*)
  008:   1 (*global*)      1 (*global*)       3 (GLIBCXX_3.4)   2 (GLIBC_2.2.5)
  00c:   3 (GLIBCXX_3.4)   3 (GLIBCXX_3.4)

Version needs section '.gnu.version_r' contains 2 entries:
 Addr: 0x00000000000006a0  Offset: 0x0006a0  Link: 7 (.dynstr)
  000000: Version: 1  File: libstdc++.so.6  Cnt: 1
  0x0010:   Name: GLIBCXX_3.4  Flags: none  Version: 3
  0x0020: Version: 1  File: libc.so.6  Cnt: 3
  0x0030:   Name: GLIBC_2.4  Flags: none  Version: 5
  0x0040:   Name: GLIBC_2.34  Flags: none  Version: 4
  0x0050:   Name: GLIBC_2.2.5  Flags: none  Version: 2

Displaying notes found in: .note.gnu.property
  Owner                Data size        Description
  GNU                  0x00000020       NT_GNU_PROPERTY_TYPE_0
      Properties: x86 feature: IBT, SHSTK
        x86 ISA needed: x86-64-baseline

Displaying notes found in: .note.gnu.build-id
  Owner                Data size        Description
  GNU                  0x00000014       NT_GNU_BUILD_ID (unique build ID bitstring)
    Build ID: c605f42bc27185ef5dd7abb366bf9b805c30333b

Displaying notes found in: .note.ABI-tag
  Owner                Data size        Description
  GNU                  0x00000010       NT_GNU_ABI_TAG (ABI version tag)
    OS: Linux, ABI: 3.2.0
```

使用objdump命令：

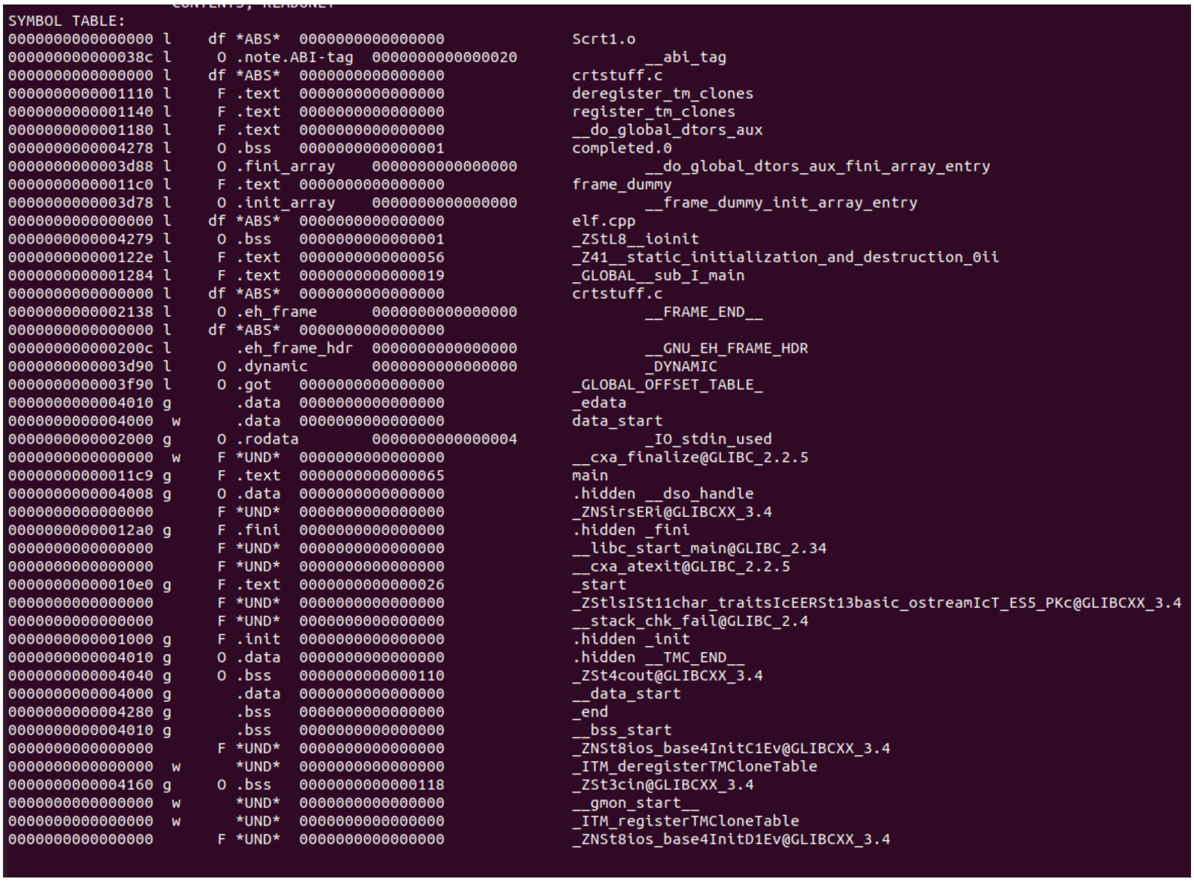
objdump -x elf



```
gdz@gdz-virtual-machine:~$ objdump -x elf

elf:     file format elf64-x86-64
elf
architecture: i386:x86-64, flags 0x00000150:
HAS_SYMS, DYNAMIC, D_PAGED
start address 0x00000000000010e0

Program Header:
    PHDR off    0x0000000000000040 vaddr 0x0000000000000040 paddr 0x0000000000000040 align 2**3
         filesz 0x00000000000002d8 memsz 0x00000000000002d8 flags r--
  INTERP off    0x0000000000000318 vaddr 0x0000000000000318 paddr 0x0000000000000318 align 2**0
         filesz 0x000000000000001c memsz 0x000000000000001c flags r--
    LOAD off    0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**12
         filesz 0x0000000000000898 memsz 0x0000000000000898 flags r--
    LOAD off    0x0000000000001000 vaddr 0x0000000000001000 paddr 0x0000000000001000 align 2**12
         filesz 0x00000000000002ad memsz 0x00000000000002ad flags r-x
    LOAD off    0x0000000000002000 vaddr 0x0000000000002000 paddr 0x0000000000002000 align 2**12
         filesz 0x000000000000013c memsz 0x000000000000013c flags r--
    LOAD off    0x0000000000002d78 vaddr 0x0000000000003d78 paddr 0x0000000000003d78 align 2**12
         filesz 0x0000000000000298 memsz 0x0000000000000508 flags rw-
 DYNAMIC off    0x0000000000002d90 vaddr 0x0000000000003d90 paddr 0x0000000000003d90 align 2**3
         filesz 0x0000000000000200 memsz 0x0000000000000200 flags rw-
    NOTE off    0x0000000000000338 vaddr 0x0000000000000338 paddr 0x0000000000000338 align 2**3
         filesz 0x0000000000000030 memsz 0x0000000000000030 flags r--
    NOTE off    0x0000000000000368 vaddr 0x0000000000000368 paddr 0x0000000000000368 align 2**2
         filesz 0x0000000000000044 memsz 0x0000000000000044 flags r--
0x6474e553 off    0x0000000000000338 vaddr 0x0000000000000338 paddr 0x0000000000000338 align 2**3
         filesz 0x0000000000000030 memsz 0x0000000000000030 flags r--
EH_FRAME off    0x000000000000200c vaddr 0x000000000000200c paddr 0x000000000000200c align 2**2
         filesz 0x0000000000000044 memsz 0x0000000000000044 flags r--
   STACK off    0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**4
         filesz 0x0000000000000000 memsz 0x0000000000000000 flags rw-
   RELRO off    0x0000000000002d78 vaddr 0x0000000000003d78 paddr 0x0000000000003d78 align 2**0
         filesz 0x0000000000000288 memsz 0x0000000000000288 flags r--

Dynamic Section:
  NEEDED               libstdc++.so.6
  NEEDED               libc.so.6
  INIT                 0x0000000000001000
  FINI                 0x00000000000012a0
  INIT_ARRAY           0x0000000000003d78
  INIT_ARRAYSZ         0x0000000000000010
  FINI_ARRAY           0x0000000000003d88
  FINI_ARRAYSZ         0x0000000000000008
  GNU_HASH             0x00000000000003b0
  STRTAB               0x0000000000000530
  SYMTAB               0x00000000000003e0
  STRSZ                0x0000000000000151
  SYMENT               0x0000000000000018
  DEBUG                0x0000000000000000
  PLTGOT               0x0000000000003f90
  PLTRELSZ             0x0000000000000078
  PLTREL               0x0000000000000007
  JMPREL               0x0000000000000820
  RELA                 0x0000000000000700
  RELASZ               0x0000000000000120
  RELAENT              0x0000000000000018
  FLAGS                0x0000000000000008
```

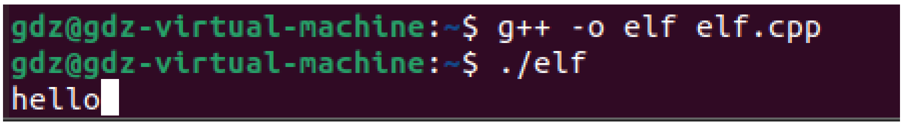


```
Version References:
  required from libstdc++.so.6:
    0x08922974 0x00 03 GLIBCXX_3.4
  required from libc.so.6:
    0x0d696914 0x00 05 GLIBC_2.4
    0x069691b4 0x00 04 GLIBC_2.34
    0x09691a75 0x00 02 GLIBC_2.2.5

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .interp       0000001c  0000000000000318  0000000000000318  00000318  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.gnu.property 00000030  0000000000000338  0000000000000338  00000338  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .note.gnu.build-id 00000024  0000000000000368  0000000000000368  00000368  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .note.ABI-tag 00000020  000000000000038c  000000000000038c  0000038c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .gnu.hash     00000030  00000000000003b0  00000000000003b0  000003b0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynsym       00000150  00000000000003e0  00000000000003e0  000003e0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .dynstr       00000151  0000000000000530  0000000000000530  00000530  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .gnu.version  0000001c  0000000000000682  0000000000000682  00000682  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .gnu.version_r 00000060  00000000000006a0  00000000000006a0  000006a0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .rela.dyn     00000120  0000000000000700  0000000000000700  00000700  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 10 .rela.plt     00000078  0000000000000820  0000000000000820  00000820  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 11 .init         0000001b  0000000000001000  0000000000001000  00001000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 12 .plt          00000060  0000000000001020  0000000000001020  00001020  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 13 .plt.got      00000010  0000000000001080  0000000000001080  00001080  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 14 .plt.sec      00000050  0000000000001090  0000000000001090  00001090  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 15 .text         000001bd  00000000000010e0  00000000000010e0  000010e0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 16 .fini         0000000d  00000000000012a0  00000000000012a0  000012a0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 17 .rodata       0000000a  0000000000002000  0000000000002000  00002000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 18 .eh_frame_hdr 00000044  000000000000200c  000000000000200c  0000200c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 19 .eh_frame     000000ec  0000000000002050  0000000000002050  00002050  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 20 .init_array   00000010  0000000000003d78  0000000000003d78  00002d78  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 21 .fini_array   00000008  0000000000003d88  0000000000003d88  00002d88  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 22 .dynamic      00000200  0000000000003d90  0000000000003d90  00002d90  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 23 .got          00000070  0000000000003f90  0000000000003f90  00002f90  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 24 .data         00000010  0000000000004000  0000000000004000  00003000  2**3
                  CONTENTS, ALLOC, LOAD, DATA
 25 .bss          00000240  0000000000004040  0000000000004040  00003010  2**6
                  ALLOC
 26 .comment      00000026  0000000000000000  0000000000000000  00003010  2**0
                  CONTENTS, READONLY
```



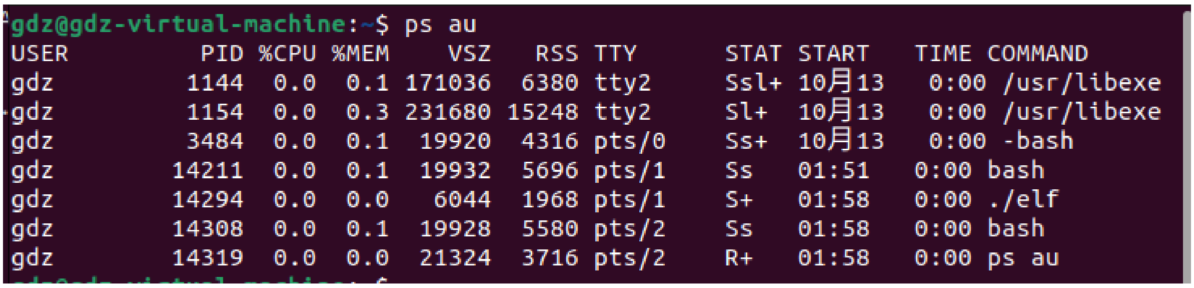
```
                  CONTENTS, READONLY
SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000              Scrt1.o
000000000000038c l     O .note.ABI-tag  0000000000000020              __abi_tag
0000000000000000 l    df *ABS*  0000000000000000              crtstuff.c
0000000000001110 l     F .text  0000000000000000              deregister_tm_clones
0000000000001140 l     F .text  0000000000000000              register_tm_clones
0000000000001180 l     F .text  0000000000000000              __do_global_dtors_aux
0000000000004278 l     O .bss   0000000000000001              completed.0
0000000000003d88 l     O .fini_array    0000000000000000              __do_global_dtors_aux_fini_array_entry
00000000000011c0 l     F .text  0000000000000000              frame_dummy
0000000000003d78 l     O .init_array    0000000000000000              __frame_dummy_init_array_entry
0000000000000000 l    df *ABS*  0000000000000000              elf.cpp
0000000000004279 l     O .bss   0000000000000001              _ZStL8__ioinit
000000000000122e l     F .text  0000000000000056              _Z41__static_initialization_and_destruction_0ii
0000000000001284 l     F .text  0000000000000019              _GLOBAL__sub_I_main
0000000000000000 l    df *ABS*  0000000000000000              crtstuff.c
0000000000002138 l     O .eh_frame      0000000000000000              __FRAME_END__
0000000000000000 l    df *ABS*  0000000000000000
000000000000200c l       .eh_frame_hdr  0000000000000000              __GNU_EH_FRAME_HDR
0000000000003d90 l     O .dynamic       0000000000000000              _DYNAMIC
0000000000003f90 l     O .got   0000000000000000              _GLOBAL_OFFSET_TABLE_
0000000000004010 g       .data  0000000000000000              _edata
0000000000004000  w      .data  0000000000000000              data_start
0000000000002000 g     O .rodata        0000000000000004              _IO_stdin_used
0000000000000000  w    F *UND*  0000000000000000              __cxa_finalize@GLIBC_2.2.5
00000000000011c9 g     F .text  0000000000000065              main
0000000000004008 g     O .data  0000000000000000              .hidden __dso_handle
0000000000000000       F *UND*  0000000000000000              _ZNSirsERi@GLIBCXX_3.4
00000000000012a0 g     F .fini  0000000000000000              .hidden _fini
0000000000000000       F *UND*  0000000000000000              __libc_start_main@GLIBC_2.34
0000000000000000       F *UND*  0000000000000000              __cxa_atexit@GLIBC_2.2.5
00000000000010e0 g     F .text  0000000000000026              _start
0000000000000000       F *UND*  0000000000000000              _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@GLIBCXX_3.4
0000000000000000       F *UND*  0000000000000000              __stack_chk_fail@GLIBC_2.4
0000000000001000 g     F .init  0000000000000000              .hidden _init
0000000000004010 g     O .data  0000000000000000              .hidden __TMC_END__
0000000000004040 g     O .bss   0000000000000110              _ZSt4cout@GLIBCXX_3.4
0000000000004000 g       .data  0000000000000000              __data_start
0000000000004280 g       .bss   0000000000000000              _end
0000000000004010 g       .bss   0000000000000000              __bss_start
0000000000000000       F *UND*  0000000000000000              _ZNSt8ios_base4InitC1Ev@GLIBCXX_3.4
0000000000000000  w      *UND*  0000000000000000              _ITM_deregisterTMCloneTable
0000000000004160 g     O .bss   0000000000000118              _ZSt3cin@GLIBCXX_3.4
0000000000000000  w      *UND*  0000000000000000              __gmon_start__
0000000000000000  w      *UND*  0000000000000000              _ITM_registerTMCloneTable
0000000000000000       F *UND*  0000000000000000              _ZNSt8ios_base4InitD1Ev@GLIBCXX_3.4
```

编译运行elf.c



```
gdz@gdz-virtual-machine:~$ g++ -o elf elf.cpp
gdz@gdz-virtual-machine:~$ ./elf
hello
```

开启另一个终端，查看PID

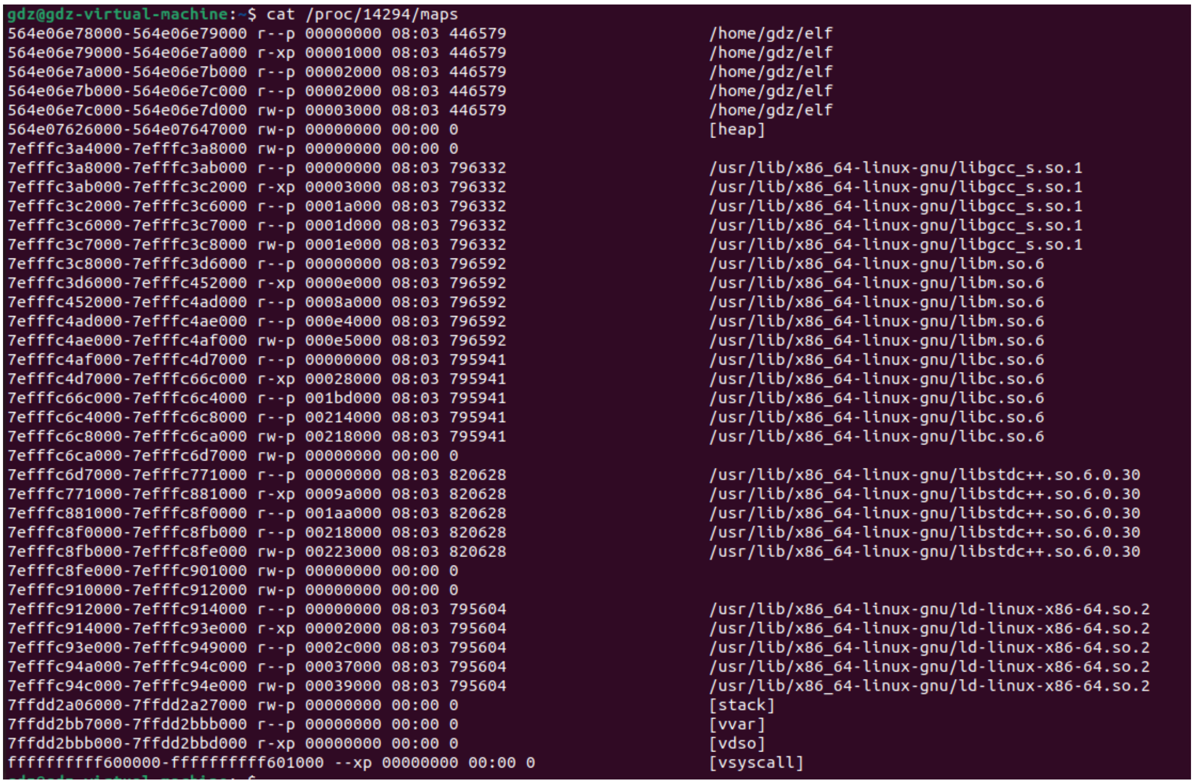


```
gdz@gdz-virtual-machine:~$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
gdz         1144  0.0  0.1 171036  6380 tty2     Ssl+ 10月13   0:00 /usr/libexe
gdz         1154  0.0  0.3 231680 15248 tty2     Sl+  10月13   0:00 /usr/libexe
gdz         3484  0.0  0.1  19920  4316 pts/0    Ss+  10月13   0:00 -bash
gdz        14211  0.0  0.1  19932  5696 pts/1    Ss   01:51   0:00 bash
gdz        14294  0.0  0.0   6044  1968 pts/1    S+   01:58   0:00 ./elf
gdz        14308  0.0  0.1  19928  5580 pts/2    Ss   01:58   0:00 bash
gdz        14319  0.0  0.0  21324  3716 pts/2    R+   01:58   0:00 ps au
gdz@gdz-virtual-machine:~$
```

PID为14294

查看memory layout.

`cat /proc/14294/maps`



```
gdz@gdz-virtual-machine:~$ cat /proc/14294/maps
564e06e78000-564e06e79000 r--p 00000000 08:03 446579                     /home/gdz/elf
564e06e79000-564e06e7a000 r-xp 00001000 08:03 446579                     /home/gdz/elf
564e06e7a000-564e06e7b000 r--p 00002000 08:03 446579                     /home/gdz/elf
564e06e7b000-564e06e7c000 r--p 00002000 08:03 446579                     /home/gdz/elf
564e06e7c000-564e06e7d000 rw-p 00003000 08:03 446579                     /home/gdz/elf
564e07626000-564e07647000 rw-p 00000000 00:00 0                          [heap]
7efffc3a4000-7efffc3a8000 rw-p 00000000 00:00 0
7efffc3a8000-7efffc3ab000 r--p 00000000 08:03 796332                     /usr/lib/x86_64-linux-gnu/libgcc_s.so.1
7efffc3ab000-7efffc3c2000 r-xp 00003000 08:03 796332                     /usr/lib/x86_64-linux-gnu/libgcc_s.so.1
7efffc3c2000-7efffc3c6000 r--p 0001a000 08:03 796332                     /usr/lib/x86_64-linux-gnu/libgcc_s.so.1
7efffc3c6000-7efffc3c7000 r--p 0001d000 08:03 796332                     /usr/lib/x86_64-linux-gnu/libgcc_s.so.1
7efffc3c7000-7efffc3c8000 rw-p 0001e000 08:03 796332                     /usr/lib/x86_64-linux-gnu/libgcc_s.so.1
7efffc3c8000-7efffc3d6000 r--p 00000000 08:03 796592                     /usr/lib/x86_64-linux-gnu/libm.so.6
7efffc3d6000-7efffc452000 r-xp 0000e000 08:03 796592                     /usr/lib/x86_64-linux-gnu/libm.so.6
7efffc452000-7efffc4ad000 r--p 0008a000 08:03 796592                     /usr/lib/x86_64-linux-gnu/libm.so.6
7efffc4ad000-7efffc4ae000 r--p 000e4000 08:03 796592                     /usr/lib/x86_64-linux-gnu/libm.so.6
7efffc4ae000-7efffc4af000 rw-p 000e5000 08:03 796592                     /usr/lib/x86_64-linux-gnu/libm.so.6
7efffc4af000-7efffc4d7000 r--p 00000000 08:03 795941                     /usr/lib/x86_64-linux-gnu/libc.so.6
7efffc4d7000-7efffc66c000 r-xp 00028000 08:03 795941                     /usr/lib/x86_64-linux-gnu/libc.so.6
7efffc66c000-7efffc6c4000 r--p 001bd000 08:03 795941                     /usr/lib/x86_64-linux-gnu/libc.so.6
7efffc6c4000-7efffc6c8000 r--p 00214000 08:03 795941                     /usr/lib/x86_64-linux-gnu/libc.so.6
7efffc6c8000-7efffc6ca000 rw-p 00218000 08:03 795941                     /usr/lib/x86_64-linux-gnu/libc.so.6
7efffc6ca000-7efffc6d7000 rw-p 00000000 00:00 0
7efffc6d7000-7efffc771000 r--p 00000000 08:03 820628                     /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
7efffc771000-7efffc881000 r-xp 0009a000 08:03 820628                     /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
7efffc881000-7efffc8f0000 r--p 001aa000 08:03 820628                     /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
7efffc8f0000-7efffc8fb000 r--p 00218000 08:03 820628                     /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
7efffc8fb000-7efffc8fe000 rw-p 00223000 08:03 820628                     /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30
7efffc8fe000-7efffc901000 rw-p 00000000 00:00 0
7efffc910000-7efffc912000 rw-p 00000000 00:00 0
7efffc912000-7efffc914000 r--p 00000000 08:03 795604                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7efffc914000-7efffc93e000 r-xp 00002000 08:03 795604                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7efffc93e000-7efffc949000 r--p 0002c000 08:03 795604                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7efffc94a000-7efffc94c000 r--p 00037000 08:03 795604                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7efffc94c000-7efffc94e000 rw-p 00039000 08:03 795604                     /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ffdd2a06000-7ffdd2a27000 rw-p 00000000 00:00 0                          [stack]
7ffdd2bb7000-7ffdd2bbb000 r--p 00000000 00:00 0                          [vvar]
7ffdd2bbb000-7ffdd2bbd000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
gdz@gdz-virtual-machine:~$
```