Lab1

lab1-1

分成 exam 和 Extra ，主要考察的还是 elf 文件格式。

粗略画了下 elf 文件的示意图，其中 {} 表示结构体，[] 表示数组

```
|--------------------
|elf_header{
|       section_header_table_offset
|       program_header_table_offset
|       ...
|}
|--------------------
|       ...
|--------------------
|       section header table:
|               an array of section header{}
|       section header-1 {
|               section addr
|               ...
|       }
|
|       section header-2 {
|               section addr
|       }
|
|       ...
|--------------------
```

exam

lab1 课下，只需将 section addr 输出，课上，要将 program header 里的 offset 和 align 输出。

```
int readelf(u_char *binary, int size)
{
    Elf32_Ehdr *ehdr = (Elf32_Ehdr *)binary;

    int Nr;

    Elf32_Phdr *phdr = NULL;

    u_char *ptr_ph_table = NULL;
    Elf32_Half ph_entry_count;
    Elf32_Half ph_entry_size;

    // check whether `binary` is a ELF file.
    if (size < 4 || !is_elf_format(binary))
    {
        printf("not a standard elf format\n");
        return 0;
    }

    // get section table addr, section header number and section header size.
    ptr_ph_table = ehdr->e_phoff + binary;
    ph_entry_count = ehdr->e_phnum;
    ph_entry_size = ehdr->e_phentsize;

    // for each section header, output section number and section addr.
    // hint: section number starts at 0.
    for (Nr = 0; Nr < ph_entry_count; Nr++)
    {
        phdr = (Elf32_Phdr *)(ptr_ph_table + ph_entry_size * Nr);
        printf("%d:0x%x,0x%x\n", Nr, phdr->p_offset, phdr->p_align);
    }

    return 0;
}
```

Extra

题目

我们知道，数据的存储分为大小端，并且 elf header 中的 e_ident[5] 指明了文件中的数据编码格式，你也可以在 ELF 手册中找到相关的信息。

现在，要你修改 readelf.c 文件，

当其解析大端存储的 elf 格式文件时，输出每个 section 的 addr

当其解析小端存储的文件时，输出每个 program 的 filesz 和 memsz

gexmul 文件夹内的 vmlinux 文件以大端存储， readelf 文件夹内 testELF 文件以小端存储，它们可以用于测试你改写的 readelf 文件的正确性。

解决

难点有两个，一是大端存储怎么转换为小端存储，二是要把哪些数据从大端转换成小端。

先简单讲下大小端，大端`0x12_34_56_78`，对应小端就是`0x78_56_34_12`，将四个字节 reverse 即可。

对于难点一：

```
#define REVERSE_32(n) \
    ((((n)&0xff) << 24) | (((n)&0xff00) << 8) | (((n) >> 8) & 0xff00) | (((n) >>
24) & 0xff))

#define REVERSE_16(n) \
    ((((n)&0xff) << 8) | (((n) >> 8) & 0xff))
```

就是拿到一个 32 位的 integer，通过位运算，取得想要的字节，再左右移动，使得顺序改变即可。

或是如此:

```
uint32_t rvs_32(uint32_t *ptr)
{
    u_char char1, char2, char3, char4;
    char1 = *((u_char *)ptr);
    char2 = *((u_char *)ptr + 1);
    char3 = *((u_char *)ptr + 2);
    char4 = *((u_char *)ptr + 3);
    return ((uint32_t)char1 << 24) + ((uint32_t)char2 << 16) + ((uint32_t)char3
<< 8) + ((uint32_t)char4);
}

uint16_t rvs_16(uint16_t *ptr)
{
    u_char char1, char2;
    char1 = *((u_char *)ptr);
    char2 = *((u_char *)ptr + 1);
    // char3 = *((U_char*) ptr + 2);
    // cahr4 = *((u_char*) ptr + 3);
    return ((uint16_t)char1 << 8) + ((uint16_t)char2);
}
```

先给个指向 32bit 的指针，把指针给强转成指向 8bit，然后把这 8bit 一个一个通过 * 运算符拿出来，最后把 8bit 强转为 32bit ，拼接好就行。

第二个难点——哪些数据需要反转

或许有人说，输出地址，那直接把地址反转就好了，其实不然。

首先要通过`offset`拿到`program header table`的地址时，`offset`需要翻转。

遍历每个`segement`时,需要`count`和`size`，这个也需要翻转。

所以，对于本次`Extra`，有四处需要翻转。

```
int readelf(u_char *binary, int size)
```

```
{
    Elf32_Ehdr *ehdr = (Elf32_Ehdr *)binary;

    int Nr;
    char ident;
    ident = ehdr->e_ident[5];

    // check whether `binary` is a ELF file.
    if (size < 4 || !is_elf_format(binary))
    {
        printf("not a standard elf format\n");
        return 0;
    }

    Elf32_Phdr *phdr = NULL;
    u_char *ptr_ph_table = NULL;
    Elf32_Half ph_entry_count;
    Elf32_Half ph_entry_size;

    if (ident == 1)
    { // little endian

        // get section table addr, section header number and section header
size.
        ptr_ph_table = ehdr->e_phoff + binary;
        ph_entry_count = ehdr->e_phnum;
        ph_entry_size = ehdr->e_phentsize;

        // for each section header, output section number and section addr.
        // hint: section number starts at 0.
        for (Nr = 0; Nr < ph_entry_count; Nr++)
        {
            phdr = (Elf32_Phdr *)(ptr_ph_table + ph_entry_size * Nr);
            printf("%d:0x%x,0x%x\n", Nr, phdr->p_filesz, phdr->p_memsz);
        }
    }
    else
    { // big endian

        // get section table addr, section header number and section header
size.
        ptr_sh_table = r_32(ehdr->e_shoff) + binary;
        sh_entry_count = r_16((ehdr->e_shnum));
        sh_entry_size = r_16((ehdr->e_shentsize));

        // for each section header, output section number and section addr.
        // hint: section number starts at 0.
        for (Nr = 0; Nr < sh_entry_count; Nr++)
        {
            shdr = (Elf32_Shdr *)(ptr_sh_table + sh_entry_size * Nr);
            printf("%d:0x%x\n", Nr, r_32((shdr->sh_addr)));
        }
    }
    return 0;
}
```

lab1-2

建议

我个人会在桌面上新建一个 txt，用来存放：

1. 常用的命令，如`gxemul -E testmips -C R3000 -M 64 vmlinux`和`git push ...`
2. 虚拟机上编译报的错，一是方便查看哪错了，二是 lab 挂了可以粘贴到虚拟机的文件带回课下研究

我还会同时打开`dev c++`，好处有两个：

1. 把自己的一些代码粘贴进去，方便自己看
2. 写一些超级小的程序，一般不超过 20 行，测试下自己的思路

对了，有的同学会把`gxemul -E testmips -C R3000 -M 64 ./gxemul/vmlinux`、`git add .`和`git commit -a -m "sth"`直接写进 makefile，非常 nice。

exam_2021

题目

我们已经补全了 print 的代码，现在有俩结构体：

```
struct s1
{
    int a;
    char b;
    char c;
    int d;
};

struct s2
{
    int size;
    int c[c_size];
};
```

说明：

1. `struct s2`里，`c_size`保证和`size`相等
2. 给`printf`传入的相关参数是`struct`的**地址**

   `printf("%$1T", addr_of_s1)`，输出第一个结构体中的内容，格式如`{1,"a","b",2}`

   `printf("%$2T", addr_of_s2)`，输出第二个结构体中的内容，格式如`{4,3,1,2,4}`。也就是说，size 和数组里面的值要一起输出。

解答

1. 全局定义如下两个结构体。

```
struct s1
{
    int a;
    char b;
    char c;
    int d;
};

struct s2
{
    int size;
    int c[];
};
```

2. 声明变量中添加如下声明。

```
struct s1 *st1;
struct s2 *st2;
int type;
```

3. 判断符号'$'的时候记得先给 type 赋初值。

```
type = 0;
if (*fmt == '$')
{
    fmt++;
    type = Ctod(*fmt), fmt++;
}
```

4. switch 当中添加 case 'T' 部分

```
case 'T':
if (type == 1)
{
    st1 = va_arg(ap, struct s1 *);

    length = PrintChar(buf, '{', 1, 0);
    OUTPUT(arg, buf, length);

    negFlag = 0;
    int a = st1->a;
    if (a < 0)
    {
        a = -a;
        negFlag = 1;
    }
    length = PrintNum(buf, a, 10, negFlag, width, ladjust, padc, 0);
    OUTPUT(arg, buf, length);
    length = PrintChar(buf, ',', 1, 0);
    OUTPUT(arg, buf, length);

    char b = st1->b;
    length = PrintChar(buf, b, width, ladjust);
```

```
    OUTPUT(arg, buf, length);
    length = PrintChar(buf, ',', 1, 0);
    OUTPUT(arg, buf, length);

    char c0 = st1->c;
    length = PrintChar(buf, c0, width, ladjust);
    OUTPUT(arg, buf, length);
    length = PrintChar(buf, ',', 1, 0);
    OUTPUT(arg, buf, length);

    int dd = st1->d;
    negFlag = 0;
    if (dd < 0)
    {
        dd = -dd;
        negFlag = 1;
    }
    length = PrintNum(buf, dd, 10, negFlag, width, ladjust, padc, 0);
    OUTPUT(arg, buf, length);

    length = PrintChar(buf, '}', 1, 0);
    OUTPUT(arg, buf, length);
}
else
{
    st2 = va_arg(ap, struct s2 *);
    length = PrintChar(buf, '{', 1, 0);
    OUTPUT(arg, buf, length);

    int size = st2->size;
    negFlag = 0;
    if (size < 0)
    {
        size = -size;
        negFlag = 1;
    }
    length = PrintNum(buf, size, 10, negFlag, width, ladjust, padc, 0);
    OUTPUT(arg, buf, length);
    if (size == 0)
    {
        length = PrintChar(buf, '}', 1, 0);
        OUTPUT(arg, buf, length);
    }
    else
    {
        length = PrintChar(buf, ',', 1, 0);
        OUTPUT(arg, buf, length);
    }
    int for_i = 0;
    int *array = st2->c;
    for (for_i = 0; for_i < size; for_i++)
    {
        int temp = array[for_i];
        negFlag = 0;
        if (temp < 0)
```

```
            {
                temp = -temp;
                negFlag = 1;
            }
            length = PrintNum(buf, temp, 10, negFlag, width, ladjust, padc, 0);
            OUTPUT(arg, buf, length);
            if (for_i != size - 1)
            {
                length = PrintChar(buf, ',', 1, 0);
                OUTPUT(arg, buf, length);
            }
            else
            {
                length = PrintChar(buf, '}', 1, 0);
                OUTPUT(arg, buf, length);
            }
        }
    }
    break;
```

测试

测试比较简单，在`init`目录下的`init.c`文件中，声明struct，初始化后输出就行。

```
typedef struct
{
    int a;
    char b;
    char c;
    int d;
} s1;

typedef struct
{
    int size;
    int c[100];
} s2;

void mips_init()
{
    printf("init.c:\tmips_init() is called\n");

    s2 s = {3, {1, 2, 3}};
    printf("%$2T", &s);
    printf("\n");
}
```

exam_2019

看起来是打印数组内容。

与上面步骤一样。

1. 添加声明。

```
int *a_int;
long int *a_long;
int var_for;

int arraySize;
```

2. 遍历 fmt

```
/* check for arraySize */
arraySize = 0;
if (*fmt == '#')
{
    fmt++;
    while (IsDigit(*fmt))
    {
        arraySize = arraySize * 10 + Ctod(*fmt);
        fmt++;
    }
}
```

3. 修改 case 内容

```
case 'a':
case 'A':
if (longFlag)
{
    a_long = va_arg(ap, long int *);
}
else
{
    a_int = va_arg(ap, int *);
}
length = PrintChar(buf, '{', 1, 0);
OUTPUT(arg, buf, length);
for (var_for = 0; var_for < arraySize; var_for++)
{
    if (longFlag)
    {
        num = a_long[var_for];
    }
    else
    {
        num = a_int[var_for];
    }
    negFlag = 0;
    if (num < 0)
    {
        num = -num;
        negFlag = 1;
    }
    length = PrintNum(buf, num, 10, negFlag, width, ladjust, padc, 0);
    OUTPUT(arg, buf, length);
    if (var_for != arraySize - 1)
    {
```

```
        length = PrintChar(buf, ',', 1, 0);
        OUTPUT(arg, buf, length);
    }
}
length = PrintChar(buf, '}', 1, 0);
OUTPUT(arg, buf, length);
break;
```

Extra_2019

没什么好说的，不是原题直接寄。

这玩意铁铁的不会。

```
#include <asm/regdef.h>
#include <asm/cp0regdef.h>
#include <asm/asm.h>

LEAF(calculator)

next_line:

li s2, 0
li s3, 0
li s4, 0
li s5, 0

loop_begin:
    lui s1, 0xb000
    lb t1, 0x00(s1)
    beq t1, zero, loop_begin
    nop

sb t1, 0x00(s1)

li t2, 81
beq t1, t2, calc_close
nop

li t2, 43
bne t1, t2, if_else_1
nop
li s5, 1
j if_end_1
nop

if_else_1:

li t2, 45
bne t1, t2, if_else_2
nop
    li s5, -1
j if_end_2
nop
```

```
if_else_2:

li t2, 68
bne t1, t2, if_else_3
nop
    li s4, 1
j if_end_3
nop

if_else_3:

li t2, 10
bne t1, t2, if_else_4
nop

    mult s3, s5
    mflo s3
    add s0, s2, s3

    while_1:
    beq s0, zero, while_1_end
    nop
        li t3, 10
        div s0, t3
        mflo s0
        mfhi t3
        addi t3, t3, 48
        sb t3, 0x00(s1)
    j while_1
    nop

    while_1_end:
    li t3, 10
    sb t3, 0x00(s1)
    j next_line;
    nop

j if_end_4
nop

if_else_4:

bne s4, zero, second
nop
    li t3, 10
    mult s2, t3
    mflo s2
    addi t3, t1, -48
    add s2, s2, t3

j if_end_5
nop

second:
    li t3, 10
```

```
    mult s3, t3
    mflo s3
    addi t3, t1, -48
    add s3, s3, t3
if_end_5:

if_end_4:
if_end_3:
if_end_2:
if_end_1:

j   loop_begin
nop

calc_close:
sb zero, 0x10(s1)

END(calculator)
```

总结

struct的定义和初始化

然后就是如何给一个`struct`赋值。

附一个初始化实例：

```
typedef struct
{
    int a;
    int b;
} s2;

s2 s = {1, 2};
```

struct获取成员变量

时隔一年有余，早就忘记了通过`.`和`->`获取struct中成员变量的区别了。

现在告诉那些忘记了的同学，`struct.var`和`struct_pointer->var`。

怪不得我在`devc++`里面“温习”`struct`的使用时，要用`.`来获取元素值，而在`print.c`里，我最后用`->`获取元素的值。我以为二者用法是相同的，但是`struct_pointer.var`编译会报错，提交的时候误打误撞写对了。

附上一个样例：

```
#include <stdio.h>

typedef struct
{
    int size;
    int c[100];
} s2;
```

```
int main()
{
    s2 s = {1, {1, 2, 3}};
    printf("%d\n", s.c[2]);
    s2 *s_ptr = &s;
    printf("%d\n", s_ptr->size);
    return 0;
}
```

学习struct的初始化，有助于我们修改init.c，测试自己的程序，甚至猜一猜自己的程序错在哪里，不过调试是不存在的。

下main这种初始化方式是错误的：

```
#include <stdio.h>

typedef struct
{
    int size;
    int *c; // 注意这里，声明成了一个指针而不是数组
} s2;

int main()
{
    s2 s = {1, {1, 2, 3}}; // 将无法正确接受参数
    printf("%d\n", s.c[2]);
    s2 *s_ptr = &s;
    printf("%d\n", s_ptr->size);
    return 0;
}
```

诡异的是，这样写，可以编译通过，在虚拟机上跑，会输出一些奇怪的数，这也是我在本地测试时犯下的第二个错误。