Lab1实验报告

Thikning 1.1

> objdump是一个强大的二进制文件分析工具，常用于反汇编和查看目标文件的详细信息，常用的参数有以下几个：
>
> - -d：反汇编可执行段
> - -D：反汇编所有段
> - -S：在反汇编代码中混合显示源代码
> - -j：指定要反汇编的段
> - -g：显示调试信息
> - -f：显示文件头信息，包括文件类型、架构、字节序等
>
> 例如下图：

```
\git@23371389:~/23371389/tools/readelf (lab1)$ mips-linux-gnu-objdump -Sd hello.o > hello.txt
git@23371389:~/23371389/tools/readelf (lab1)$ cat hello.txt

hello.o:     文件格式 elf32-tradbigmips


Disassembly of section .text:

00000000 <main>:
   0:   27bdffe0        addiu   sp,sp,-32
   4:   afbf001c        sw      ra,28(sp)
   8:   afbe0018        sw      s8,24(sp)
   c:   03a0f025        move    s8,sp
  10:   3c1c0000        lui     gp,0x0
  14:   279c0000        addiu   gp,gp,0
  18:   afbc0010        sw      gp,16(sp)
  1c:   3c020000        lui     v0,0x0
  20:   24440000        addiu   a0,v0,0
  24:   8f820000        lw      v0,0(gp)
  28:   0040c825        move    t9,v0
  2c:   0320f809        jalr    t9
  30:   00000000        nop
  34:   8fdc0010        lw      gp,16(s8)
  38:   00001025        move    v0,zero
  3c:   03c0e825        move    sp,s8
  40:   8fbf001c        lw      ra,28(sp)
  44:   8fbe0018        lw      s8,24(sp)
  48:   27bd0020        addiu   sp,sp,32
  4c:   03e00008        jr      ra
  50:   00000000        nop
```

Thinking 1.2

```
git@23371389:~/23371389 (lab1-extra)$ ./tools/readelf/readelf target/mos
0:0x0
1:0x80020000
2:0x80021bf0
3:0x80021c08
4:0x80021c20
5:0x0
6:0x0
7:0x0
8:0x0
9:0x0
10:0x0
11:0x0
12:0x0
13:0x0
14:0x0
15:0x0
16:0x0
17:0x0
```

```
git@23371389:~/23371389/tools/readelf (lab1-extra)$ readelf -h readelf
ELF 头:
  Magic:    7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  类别:                              ELF64
  数据:                              2 补码，小端序 (little endian)
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI 版本:                          0
  类型:                              DYN (Position-Independent Executable file)
  系统架构:                          Advanced Micro Devices X86-64
  版本:                              0x1
  入口点地址:               0x1180
  程序头起点:          64 (bytes into file)
  Start of section headers:          14488 (bytes into file)
  标志:             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
```

我们的 `hello` 文件是32位的格式，而 `readelf` 则是64位的。我们打开 `readelf.c` 文件，发现其中的的数据类

型前缀都是 `ELF32` ，也正是说明了这个程序负责分析32位的 `ELF` 文件。所以它不能分析身为64位格式程序的自己。

Thinking 1.3

> 在我们的实验中，系统启动被简化成了把内核加载到指定内存位置。
>
> MIPS系统启动时首先接管的是 `bootloader`，随后 `Linker Script` 把各个节映射到对应的段上，内核文件也在这时
>
> 被加载到合适的地址空间中。
>
> 在 `Exercise 1.2` 中，我们补全了 `kernel.lds` 文件，把 `.text`、`.data`、 `.bss` 三个段映射到了合理空间。
>
> 经过 `Linker Script` 文件的引导，内核代码就会被加载到 `0x80010000` 这段地址。再通过 `ENTRY(_start)` 的入口规
>
> 定，如此便保证了我们能够跳转到内核入口

难点分析

1. `ELF` 文件结构： `ELF` 头 + 段头表 `segment` + 节头表 `section`

- 段头表 `segment`：运行时刻使用

  组成可执行文件或者可共享文件，在运行时为加载器提供信息
- 节头表 `section`：编译和链接时刻使用

  组成可重定位文件，参与可执行文件和可共享文件的链接

  `.text` 保存可执行文件的操作指令；

  `.data` 保存已初始化的全局变量和静态变量；

  `.bss` 保存未初始化的全局变量和静态变量。

2. `printk` 函数的实现

```
void printk(const char *fmt, ...)
{
 va_list ap;
 va_start(ap, fmt);
 vprintfmt(outputk, NULL, fmt, ap);
 va_end(ap);
}
```

1. `va_list`，变长参数表的变量类型；
2. `va_start(va_list ap, lastarg)`，用于初始化变长参数表的宏；
3. `va_arg(va_list ap, 类型)`，用于取变长参数表下一个参数的宏；
4. `va_end(va_list ap)`，结束使用变长参数表的宏。

注意其可扩展性，可以通过更改 `vprintfmt` 中的回调函数 `out` 实现对于其他内存流的写入

实验体会

- 指导书涉及的知识较多，初读时感觉晦涩难懂，但多读几遍，并结合实验题目后便有了很好的理解。
- 从课下和课上实验可以看到，C语言基础（尤其是指针）是非常重要的，一定要熟练掌握。
- 要善用 `make` 去构建文件，并且熟练掌握调试方法，对于上机将有很大帮助