PA2-2 实验报告

刘时宜 201180078

2022 年 4 月 16 日

目录

1 实现过程

首先，修改`testcase/Makefile`，更改程序装载地址。

然后，只需要补充`kernel`中幅值内存内容的部分。由于是对内存进行操作，第一版想用`vaddr_write`和`vaddr_read`函数进行复制，但是这两个函数应当是`NEMU`的内部功能，相当于硬件操作，`kernel`不能够调用。于是改用指针方式，逐字节复制内存内容。

```
// @ kernel/src/elf/elf.c
/* TODO: copy the segment from the ELF file to its proper memory area */
    for(addr_shift = 0; addr_shift < ph->p_filesz; addr_shift++)
    {
        // vaddr_write(ph->p_vaddr+addr_shift, 0, 1, vaddr_read(ph->p_offset+
    addr_shift, 0, 1));
        pdata = (void*)ph->p_offset+addr_shift;
        data = *pdata;
        pdata = (void*)ph->p_vaddr+addr_shift;
        *pdata = data;
    }
/* TODO: zeror the memory area [vaddr + file_sz, vaddr + mem_sz) */
    for(addr_shift = ph->p_filesz; addr_shift < ph->p_memsz; addr_shift++)
    {
        // vaddr_write(ph->p_vaddr+addr_shift, 0, 1, 0);
        pdata = (void*)ph->p_vaddr+addr_shift;
```

```
16          *pdata = 0;
17      }
```

以上代码中的pdata、data分别为char*和char类型，目的是进行逐字节的数据操作。make test_pa-2-2之后竟然十分丝滑地一次成功了。

2 遇到的bug以及解决

2.1 没有修改Makefile导致的指令错误

我采取代码下载下来，在本地编写程序并运行后再上传到服务器的方式完成PA作业。然而，应当是由于gcc版本不同的问题，本地的gcc编译时会产生 endbr32 和 endbr64 指令，均为英特尔公司后来加上的检查跳转位置是否合法的指令，在i386的实现中均不存在，因此在 Makefile 的 CFLAGS 中加入了 -fcf-protection=branch -mmanual-endbr 以取消这两条指令的自动生成。

然而，在PA服务器上的gcc却提示这两参数非法。故采取同步时不同步这两条指令的方法解决这个问题。

后果是，在本地修改的testcase/Makefile并没有同步到服务器上，导致内存加载混乱，使得程序行为与反汇编结果不相符的奇怪情形。

在qq群中求助大家后顺利解决了这个问题。这里特别感谢陈泰霖同学告知使用 NEMU 的 monitor，使用si命令加数字的方法使NEMU逐条打印执行的语句以及反汇编结果，使得我在混乱中最终定位了问题所在成功解决。

2.2 本地运行结果与服务器运行结果不一致

kernel写好后，在本地运行会出现程序死循环的情况，而在PA服务器上运行则可以通过所有testcase。首先通过反汇编结果发现PA服务器与本地编译的kernel并不一致。然后通过使用参考指令实现（__ref_）的方式发现本地可以正常运行，说明问题在于NEMU的指令实现。最终确定有问题的指令为push_i_b。

检查代码后发现问题居然是源操作数的data_size没有指定为8。修改后在服务器和本地均能成功运行。

如此低级的错误竟然能够通过所有pa2-1以及服务器上pa2-2的测试，说明测试样例确实不够完善。或许可以借鉴kernel的程序行为开发一些新的能够检测这个错误的测试样例。暂且不在此叙述，构建好后再行单独提交测试样例。

3 思考题

为什么在装载时要将内存中剩余的 memsz-filesz 字节的内容清零？

这部分内容为声明了但是没有初始化的全局变量的存储空间。将这部分内容清零，应当是 NEMU 约定所有全局变量的值均初始化为0。