U201614702 CS1607 樊俊超

PA1 必答题

Q1：

- 理解基础设施 我们通过一些简单的计算来体会简易调试器的作用. 首先作以下假设:
  - 假设你需要编译500次NEMU才能完成PA.
  - 假设这500次编译当中, 有90%的次数是用于调试.
  - 假设你没有实现简易调试器, 只能通过GDB对运行在NEMU上的客户程序进行调试. 在每一次调试中, 由于GDB不能直接观测客户程序, 你需要花费30秒的时间来从GDB中获取并分析一个信息.
  - 假设你需要获取并分析20个信息才能排除一个bug.

  那么这个学期下来, 你将会在调试上花费多少时间?

  由于简易调试器可以直接观测客户程序, 假设通过简易调试器只需要花费10秒的时间从中获取并分析相同的信息. 那么这个学期下来, 简易调试器可以帮助你节省多少调试的时间?

  事实上, 这些数字也许还是有点乐观, 例如就算使用GDB来直接调客户程序, 这些数字假设你能通过10分钟的时间排除一个bug. 如果实际上你需要在调试过程中获取并分析更多的信息, 简易调试器这一基础设施能带来的好处就更大.

A1：

500 X 90% X 30 X 20 / 60 = 4500min = 75h，根据我每天早九点到午时，午时二刻到晚十二点，中间休息一个半小时的做实验作息来看，需要将近六天时间专门用于 debug，加上偶尔幸运眷顾，玄学爆发，奥力给附体，再排除因为看文档尝试理解讲义的深刻内涵所花的时间，一周很长很长。

但有了调试器，我就有了做实验的信心，在看讲义做实验的忙碌过程中，（被迫在 run 或者 make 时）歇下来用调试器做个调试是真的舒服。调试器为什么有用，第一，它给了我调试的方向，让我不用在一个拥有数十个文件和十几个已编辑文件的文件夹下像无头苍蝇一样绕来绕去；第二，它给了我调试的信息，你想知道运行到哪里报错和怎么错，调试器已经贴心的记下了我要用的信息。

Q2：

- 查阅i386手册 理解了科学查阅手册的方法之后，请你尝试在i386手册中查阅以下问题所在的位置，把需要阅读的范围写到你的实验报告里面：
  - EFLAGS寄存器中的CF位是什么意思？
  - ModR/M字节是什么？
  - mov指令的具体格式是怎么样的？

A2：

CF（carry flag）进位标志位

ModR/M 对 opcode 进行补充，在主操作码字节后，这些指令大都是可以在内存中引用操作数的指令，参考 i386 手册

```
Most instructions that can refer to an operand in memory have an addressing
form byte following the primary opcode byte(s). This byte, called the ModR/M
byte, specifies the address form to be used. Certain encodings of the ModR/M
byte indicate a second addressing byte, the SIB (Scale Index Base) byte,
which follows the ModR/M byte and is required to fully specify the
addressing form.
```

mov 指令，参考手册

mov data

move to/from special registers

Q3:

- shell命令 完成PA1的内容之后，`nemu/` 目录下的所有.c和.h和文件总共有多少行代码？你是使用什么命令得到这个结果的？和框架代码相比，你在PA1中编写了多少行代码？

A3:

分别在 pa0 和 pa1 分支下使用 find . -name "*.c"|xargs cat|grep -v ^$|wc -l 命令以及 find . -name "*.h"|xargs cat|grep -v ^$|wc -l 统计后得到 pa0 下除空后行数为 2409+698=3107 行，pa1 下除空后行数为 2734+723=3457 行，我在 pa1 里大致编写了 350 行，但由于 pa0 没写，pa1 中有 make 后的内容，误差上下浮动 1000 行（多退少补）。

Q4：

- 使用man 打开工程目录下的 `Makefile` 文件，你会在 `CFLAGS` 变量中看到gcc的一些编译选项。请解释gcc中的 `-Wall` 和 `-Werror` 有什么作用？为什么要使用 `-Wall` 和 `-Werror` ？

A4：

-wall 打开 gcc 的所有警告，-werror 将所有的 warning 当作错误处理。是用-wall 和-werror 可以帮助排除非代码运行造成的错误，有效减少不必要的 bug。