深入理解计算机系统(CSAPP) 实验详解：BombLab

原创

Jackson1997__ 于 2020-11-04 10:51:25 发布 1375 收藏 12

分类专栏： CSAPP

本文链接：https://blog.csdn.net/qq448545478/article/details/109456833

更新历史

- `20201102` 开始更新
- `20201104` 完成

该实验的目的

该实验可以提高你阅读汇编代码的能力，以及加深对寄存器，内存和指针的理解，还有调试 `gdb` 能力。

前言

> BombLab，一个家喻户晓的实验。

我看过的教程中都是这么介绍它的，可见它的魅力和趣味性是如此之大。话不多说，接下来我们就来了解一下该实验吧！

环境搭建

环境我们参照DataLab里的环境搭建进行。

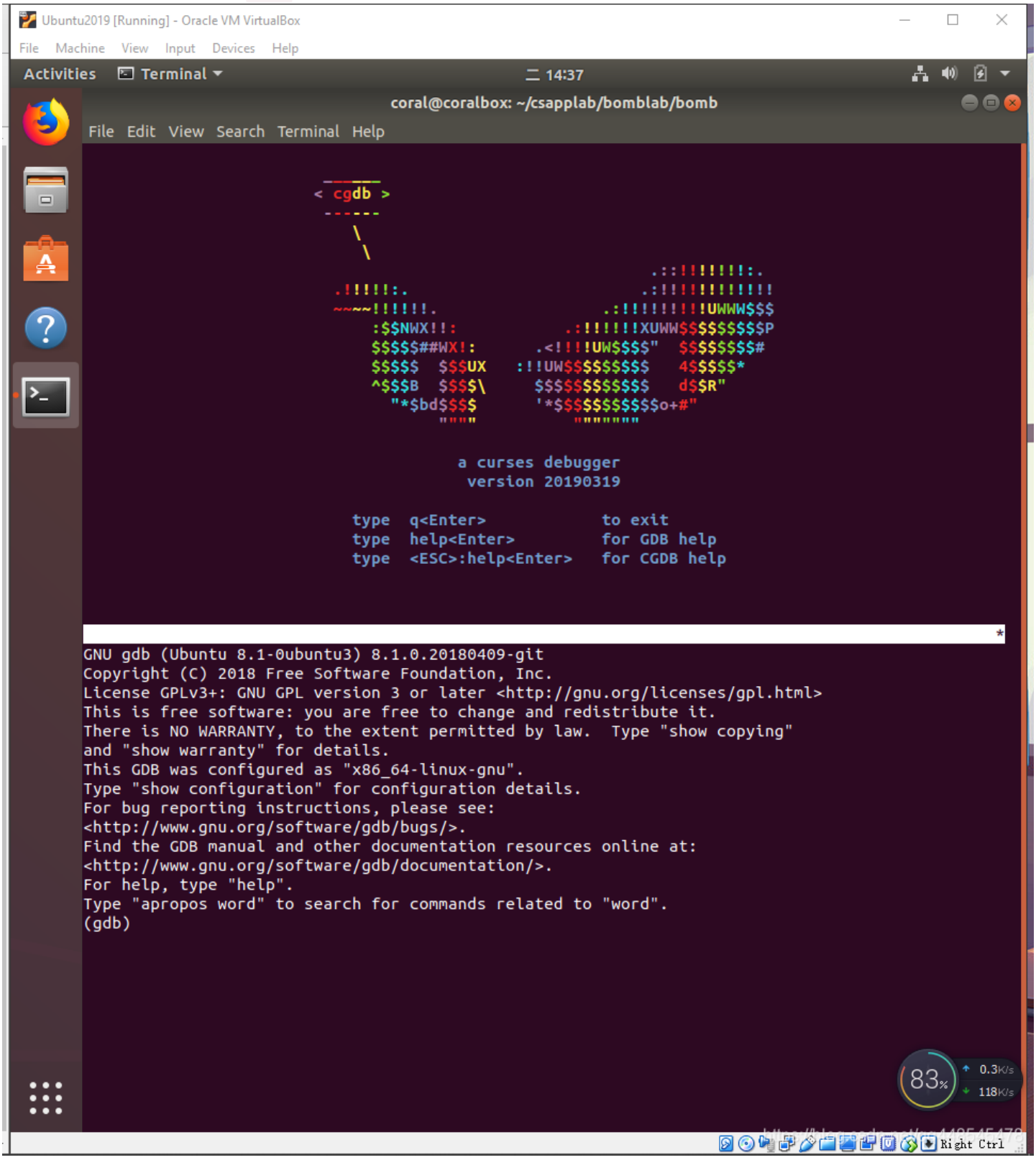
如何使用正确的姿势写代码和测试

这里需要额外安装一个环境，就是 `cgdb`，那么我们为什么不直接用gdb呢？答案就是因为cgdb更具方便在调试的时候，简单点来说有一个代码窗口可以给你看。

那么我们先从cgdb官方下载安装包，下载完后直接按照官方说明安装即可，安装遇到问题欢迎评论区，我过程中也遇到过问题，所以不要害羞~

```
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

安装完成后就可以直接终端敲 `cgdb`,进去就是一个默认界面。



你看到的可能大概率和我的不一样，因为这个图我是盗别人的-。-，差不多就行

下面讲一些基本操作

操作

整个大概的流程

在`bomb`目录打开终端，执行`cgdb bomb`，出来的界面也和上面差不多。
按`ESC`（源码窗口），然后我们按下`Ctrl+W`，切换为左右分屏模式，这一步你看你喜欢
然后在按下`i`（gdb窗口），然后在输入run，结果肯定是BOMB！！！这个是正常的，因为咋们还没有开始解谜。
后面我们会创建一个叫`in`的文件，方便我们记录密码和走流程，有了`in`文件，我们就可以把上面的`run`改为`run <in`。**当然我们要给这个`in`文件先随便输入几个字符就行了，不然会卡输入。**

常用的调试命令

- `b 函数名` 打断点在函数 例如 `b phase_1`
- `b *0x地址` 打断点在一个地址上 例如 `b *0xffffff`
- `d x` 删除断点 x是断点的索引，不输入x删除所有断点
- **`p 寄存器` 打印寄存器的值（直接引用） 例如 `p $rax`**
- **`x (寄存器)` 将寄存器的值作为内存地址，打印该地址处的值（间接引用）例如 `p ($rax)`**
- **`x (值)` 将值作为内存地址，打印该地址处的值（间接引用）例如 `p ($rax)`**
- `si` 调试单步，会进入到函数内部
- `ni` 调试单步，不会进入函数内部

打完断点后可以按`Ctrl+c`取消输入
基本上就是这些，详细的看链接

正式开始

实验开始前的一些建议

1. 如果你对汇编不熟悉，或者连CSAPP里的汇编都看不懂，建议加强一些汇编的阅读能力，因为实验的汇编代码会很复杂，不难，但是你会眼花缭乱。
2. 如果你不会用`cgdb`也没事，我一开始也不会，很简单的，操作几遍就可以很熟练的调试了
3. 建议对一大段看不懂的汇编代码进行拆分，全部写下来，化为伪代码，就马上可以看懂。

实验开始

由于我没有怎么分析，就是一顿操作就完事了，所以看大佬的文章吧，都差不多。

读厚CSAPP
CSAPP 之 Bomb Lab

建议：

第一个实验完全搞懂它，第二个实验尝试自己做，后面几个实验完全靠自己。加油！

我就是这么做的，其实都很简单，就是最后一题差点把我搞死，因为太长了，长到不想做-.-，不过最后还是坚持了下来~

总结

全部做完之后感觉自己变强了不少，不管是汇编阅读能力，还有寄存器和内存的理解都有很大的提升，总之，非常推荐大家去做。

一些汇编总结

- `leaq 指令`，做完全部实验，现在的印象里只有对内存地址的操作，就是它出现的时候一般都是把内存地址复制到寄存器中，作为寄存器的值。
- `0x4(%rax)`，有括号代表是间接引用，表示当前`寄存器值+0x4`处的`内存地址`的`值`。
- `%rax`，无括号代表直接引用，表示当前寄存器的值，不过它也有可能是个地址，所以它在进行算术操作（加减）的时候要注意它是地址，还是立即数。