Lab2 实验报告

匡院 谢逸 171240536

为了完成这个实验，你首先需要知道 glibc 库中 `rand` 和 `system` 两个函数之间地址的差距。讲义上有写用汇编来看，但是我觉得太麻烦了！我觉得可以用 printf 来直接输出函数的入口地址，所以我就写了如下的代码：

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("%u\n", rand);
    printf("%u\n", printf);
    return 0;
}

```

运行结果：

32 位：

```
xy@debian:~/lab2$ gcc -m32 a.c
xy@debian:~/lab2$ ./a.out
4149606128
4149712848
xy@debian:~/lab2$ ./a.out
4150429424
4150536144
xy@debian:~/lab2$ ./a.out
4150482672
4150589392
xy@debian:~/lab2$ ./a.out
4149626608
4149733328
xy@debian:~/lab2$ ./a.out
4149593840
4149700560
xy@debian:~/lab2$ ./a.out
4150093552
4150200272
xy@debian:~/lab2$ ./a.out
4150118128
4150224848
xy@debian:~/lab2$ ./a.out
4149696240
4149802960
```

可以发现，虽然 printf 和 rand 函数的入口地址在不断变化，但是差值是一样的！(我手算了一下)printf 的入口地址-rand 入口地址永远都是 106720！很好我们把这个数字记下来。

64 位

```
xy@debian:~/lab2$ gcc -m64 a.c
xy@debian:~/lab2$ ./a.out
335423520
335524240
xy@debian:~/lab2$ ./a.out
1903462432
1903563152
xy@debian:~/lab2$ ./a.out
4126550048
4126650768
xy@debian:~/lab2$ ./a.out
3076225056
3076325776
xy@debian:~/lab2$ ./a.out
2047060000
2047160720
xy@debian:~/lab2$ ./a.out
2987284512
2987385232
xy@debian:~/lab2$ ./a.out
3267635232
3267735952
xy@debian:~/lab2$ ./a.out
136861728
136962448
```

类似的，可以发现 printf 入口地址-rand 入口地址永远为 100720。

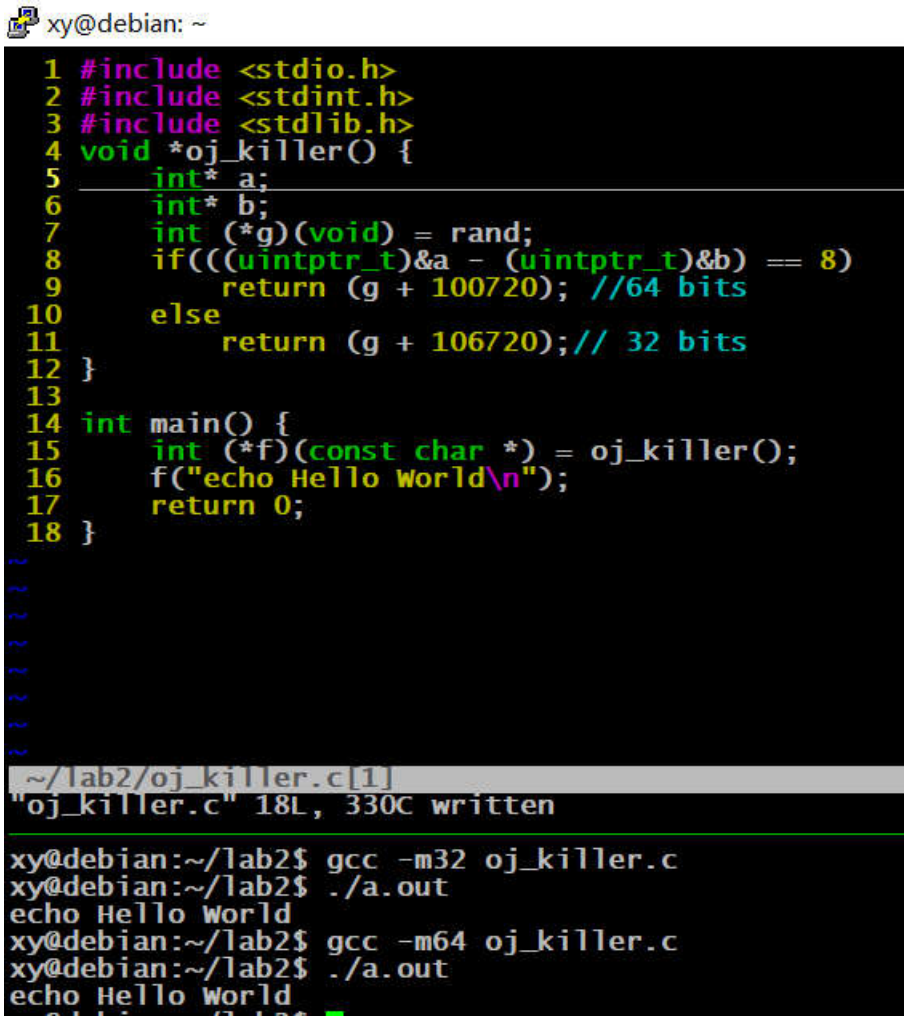
知道了差值，我们就可以写出使用于 32 位或者 64 位的代码了。比如 64 位下可以这么写：

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 void *oj_killer(){
 4     int (*g)(void) = rand;
 5     return (g + 100720);
 6 }
 7 int main(){
 8     int (*f)(const char *) = oj_killer
   ();
 9     f("echo Hello World\n");
10     return 0;
11 }
```

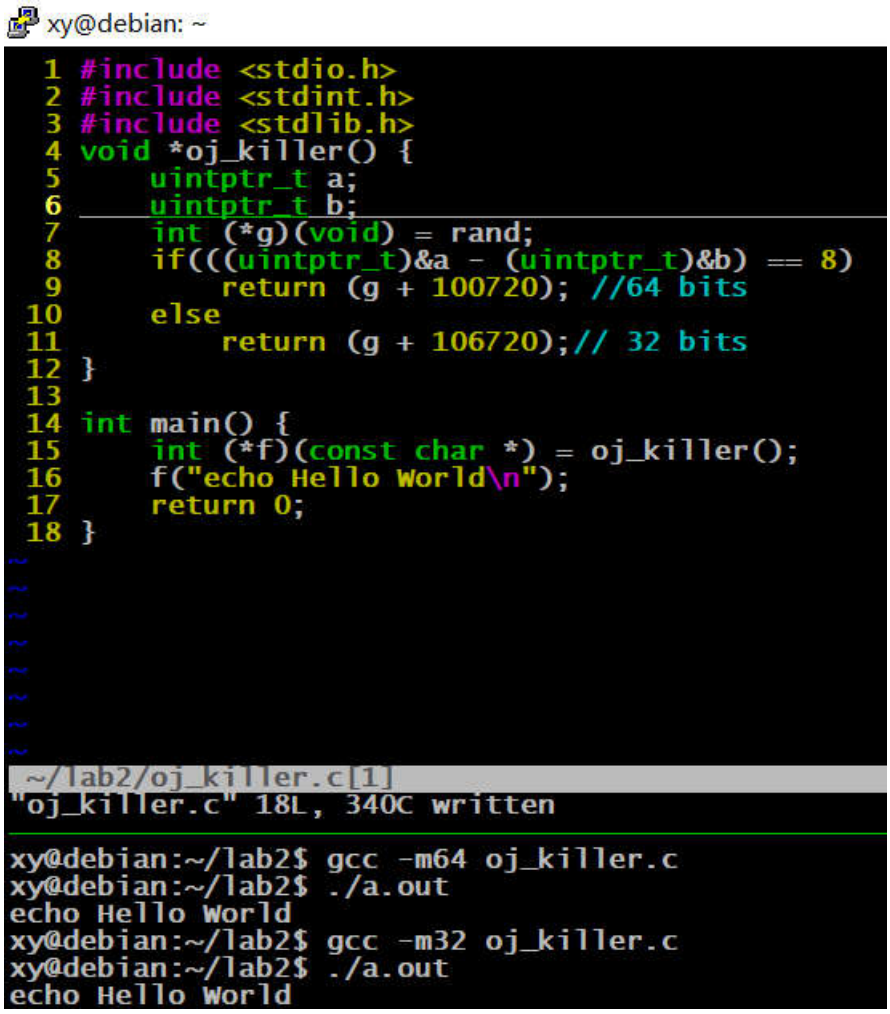
但是这样的代码在 32 位下就会 segmentation fault。

所以接下来的任务，就是判定是 32 位还是 64 位。当然这个问题实现起来也不难，你只要在运行时检查某些在 32/64 位平台下不同的数值就可以了。于是我就想到了判断指针的长度。因此我就用 sizeof()写出了第一份代码。可惜后来发现要求只允许调用一次 rand，其他的库函数都不能用，所以只能换个方法。我又想到了申明两个指针，看地址之差，不就是指针的长度了吗。但是用什么来把地址类型转换呢？我首先遵循越长越好的原则使用了 long long

int，可行，但是有个很难看的 warning，因此我觉得必须要用一个和指针长度相同的变量。因此我选择了 uintptr_t，就有了第一份符合要求的代码:

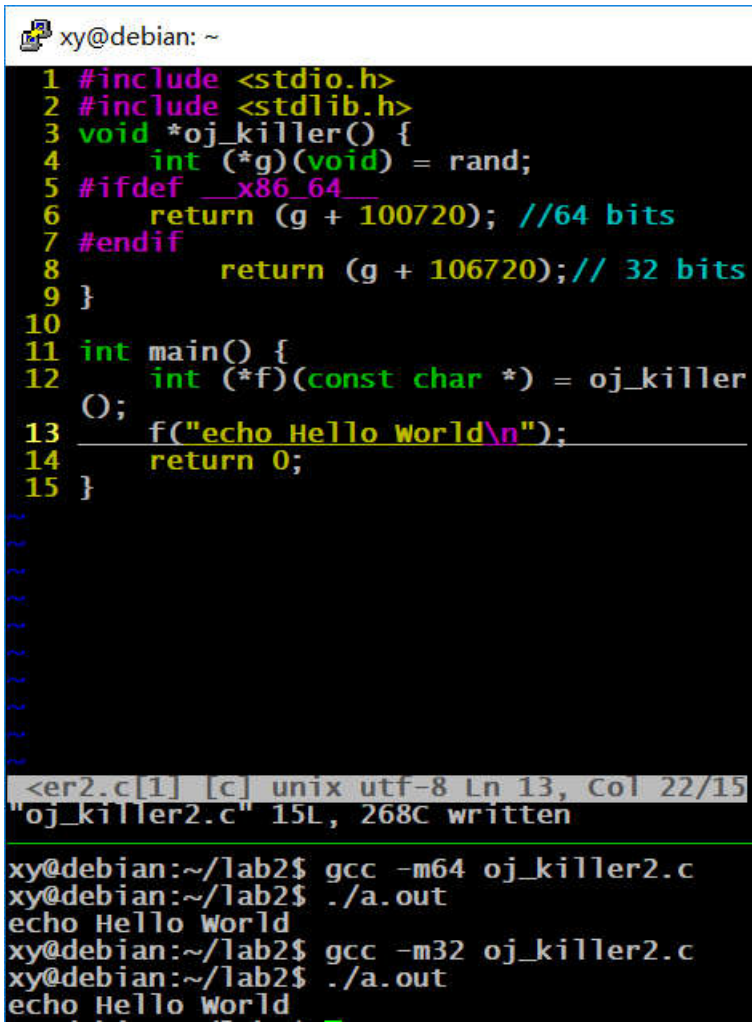


可以看到很好地兼容了 64 位和 32 位。但是既然有了 uintptr_t，我觉得干脆连指针都不用了，判断 uintptr_t 的长度就行了啊，所以就有了第二份符合要求的代码:

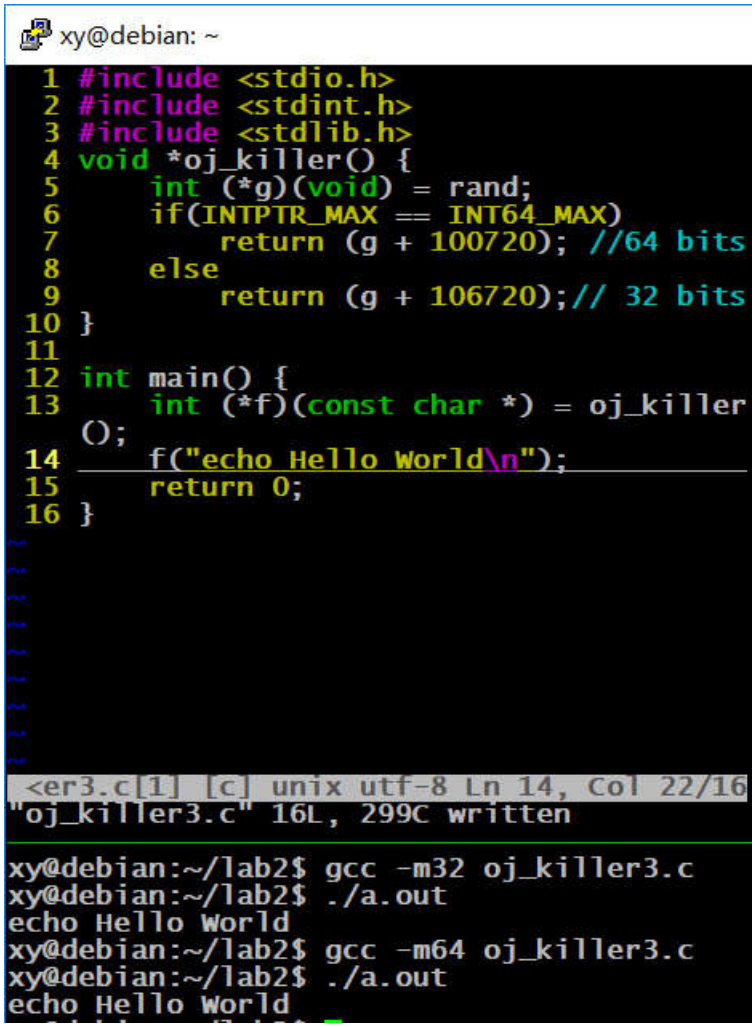


接下来，我才发现讲义上居然对判断 64 位和 32 位有提示！所以我又试了几种方法：

用`__x86_64__`宏是否定义：



用 `INTPTR_MAX` 的大小:


xy@debian: ~
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <stdlib.h>
4 void *oj_killer() {
5 int (*g)(void) = rand;
6 if(INTPTR_MAX == INT64_MAX)
7 return (g + 100720); //64 bits
8 else
9 return (g + 106720);// 32 bits
10 }
11
12 int main() {
13 int (*f)(const char *) = oj_killer
();
14 f("echo Hello World\n");
15 return 0;
16 }
<er3.c[1] [c] unix utf-8 Ln 14, Col 22/16
"oj_killer3.c" 16L, 299C written
xy@debian:~/lab2$ gcc -m32 oj_killer3.c
xy@debian:~/lab2$ ./a.out
echo Hello World
xy@debian:~/lab2$ gcc -m64 oj_killer3.c
xy@debian:~/lab2$ ./a.out
echo Hello World

报告到此结束。