2. overflow1运行结果：



1. (1) 0xbfffef1c (2) 0xbfffef24

|  |
| --- |
| return address |
| ebp |
|  |
|  |
|  |
|  |
|  |

1. 

这两条指令将立即数0x804843b放入寄存器ebp向上偏移四位的地址上，也就是原本存放foo函数返回地址的位置，而这个立即数正好是函数why\_here的开始地址，导致函数foo执行完之后继续执行了函数why\_here。

2. 16
3. 都是a
4. 0xbfffef30+N~0xbfffef30+N+3 是 ebp

0xbfffef30+N+3~0xbfffef30+N+7 存放的是函数发返回地址

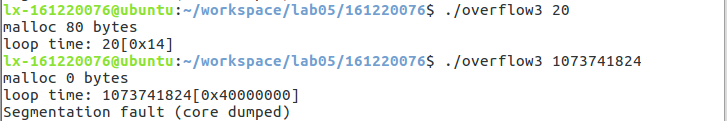
1. fgets() : 从文件结构体指针stream中读取数据，每次读取一行。读取的数据保存在buf指向的字符数组中，每次最多读取bufsize-1个字符（第bufsize个字符赋'\0'），如果文件中的该行，不足bufsize个字符，则读完该行就结束**。**所以用fgets()函数在此处不会发生覆盖ebp和函数返回地址的情况。

而gets()函数只会在读入到EOF或者换行结束。输入的字符数过多污染了ebp。

1. 栈随机化：使得栈的位置在程序每次运行时都有变化。因此，即使许多机器都运行同样的代码，它们的栈地址都是不同的。

栈破坏检测：加入栈保护着机制，当栈被破坏时，程序会自动异常退出。

2. 以不同参数的运行情况：



1. 范围为0~（1<<32）-1。第一次分配了20\*sizeof（int）个字节，第二次分配了0个字节。
2. 

在分配内存空间时，len\*sizeof(int)超出了unsigned int的表示范围，发生了溢出，溢出之后正好为0，所以分配了0个字节，在下面for循环中就会发生缓冲区溢出。

1. 为了避免由于溢出等错误而导致的程序崩溃等问题。允许分配0个字节空间就相当于没有得到空间。

时c运行库设计的问题。