程序设计实践II - 拆弹

任务概述：

实验目的

通过本阶段的任务，要求学生能够比较深入地掌握程序在计算机中是如何执行的。要求：能够看懂汇编程序，能够掌握debug调试的基本方法。

通过对一个二进制可执行程序（称为“二进制炸弹”）的理解和逆向工程，加深对对程序的机器级表示、汇编语言、调试器和逆向工程等方面知识的理解和掌握。

实验内容

- 作为实验目标的二进制炸弹“binary bombs” Linux可执行程序包含了多个阶段（或关卡），在每个阶段程序要求你输入一个特定字符串，如果输入满足程序代码所定义的要求，该阶段的炸弹就被拆除了，否则程序输出“炸弹爆炸BOOM!!! ”的提示并转到下一阶段再次等待对应的输入 ——实验的目标是设法得出解除尽可能多阶段的字符串。
- 为完成二进制炸弹拆除任务，需要通过反汇编和理解可执行炸弹文件程序或使用gdb调试器跟踪每一阶段的机器代码，从中理解关键机器指令的行为和作用，进而设法推断拆除炸弹所需的目标字符串。

实验环境

Linux 32-bit，C 汇编语言

任务1：

编写一个拆弹程序，完成如下功能，难度逐级递增：

在写完拆弹游戏后，查看自己程序对应的汇编程序，理解程序运行过程，完成文档，说明要拆解自己写的拆弹程序，应该通过查看汇编程序的，以及如何调试完成拆弹。

1. 递归调用和栈

   编写如下功能的递归程序：

$$F(n) = \begin{cases} N1, & n = 0 \\ N2, & n = 1 \\ 0.5 * F(n-2) + F(n-1) & n > 1 \end{cases}$$

   程序中调用上述递归函数的伪代码如下：

```
输入参数：input as string,包括index->n, value
输出：
if F(index)==value  拆弹成功;
else 炸弹爆炸;
```

   要求：

   - 分析自己写的程序是如何运行的，分析活动记录栈，描述程序运行过程；
   - 可以修改任务2中的bomb.c，建立自己的拆弹程序。

2. 链表/指针/结构

建立如下所示的结构体：

```
typedef struct nodeStruct
{
    int value;
    int index;
    struct nodeStruct *next;
} listNode;
```

功能：

- 输入1-7中的7个数字的某种顺序排列，要求不能超过这个范围，数字不能相同。输入的数字放在数组中。
- 已知链表如下：

```
listNode node7 = {LIST_NUMBER_7, 7, NULL};
listNode node6 = {LIST_NUMBER_6, 6, &node7};
listNode node5 = {LIST_NUMBER_5, 5, &node6};
listNode node4 = {LIST_NUMBER_4, 4, &node5};
listNode node3 = {LIST_NUMBER_3, 3, &node4};
listNode node2 = {LIST_NUMBER_2, 2, &node3};
listNode node1 = {LIST_NUMBER_1, 1, &node2};
```

其中：

LIST_NUMBER_7，。。。LIST_NUMBER_1的值是可配置的。

链表头：

```
listNode *start = &node1;
```

- 以输入数字为序号，将链表中相应序号的结构的地址记录到指针数组相应元素中。
- 以指针数组中结构的顺序重新将个结构链接成一个链表。
- 新的链表中各个结构记录的值value应该是递减排序的。

要求：

- 分析自己写的程序是如何运行的，分析链表的处理逻辑和内置链表数据，描述程序运行过程；
- 可以修改任务2中的bomb.c，建立自己的拆弹程序。

任务2：

实验数据准备：

1. **学生实验数据包：**bomblab.tar（随不同学号有所不同）**
2. **解压命令：**tar** **xvf bomblab.tar**
3. **数据包中包含下面文件：**
    - nbomb：二进制可执行炸弹程序
    - nbomb.c：bomb程序的main函数

二进制炸弹目标程序：

包含了5个阶段，分别集中考察对以下二进制程序表示各方面的理解和掌握：

1. 阶段0：字符串比较
2. 阶段1：浮点表示

3. 阶段2：循环
4. 阶段3：条件/分支
5. 阶段4：指针

bomb可执行程序接受0或1个命令行参数

1. 如果不指定参数，则bomb程序在输出欢迎信息后，等待用户按行输入每一阶段用来拆除炸弹的字符串，并根据输入字符串决定是通过相应阶段还是引爆该阶段的炸弹（输出“BOOM!!! ”）。
2. 也可将一纯文本文件作为程序的唯一参数（例如“./bomb strings.txt”），文件中的每行包含拆除对应炸弹阶段的字符串，程序将依次检查每一阶段拆除字符串的正确性来决定炸弹拆除成败。
3. 在拆除炸弹的过程中，可以选择跳过一些暂时未能拆除的阶段
4. 可在要跳过的阶段中输入任意非空白（即不全是空格、制表、换行字符）字符串，将引爆相应阶段的炸弹，但程序不会中止而是进入下一阶段。

炸弹程序运行示例：

1. **二进制炸弹程序运行示例****1 –** **未指定命令行参数：**
    1. $** **./bomb**
    2. Welcome to my fiendish little bomb. You have 5 phases with
    3. which to blow yourself up. Have a nice day!
    4. （等待输入阶段0的拆解字符串）
2. 二进制炸弹程序运行示例2 –** **以包含拆解字符串的文件作为参数：**

    $**./bomb answers.txt**

    Welcome to my fiendish little bomb. You have 5 phases with

    which to blow yourself up. Have a nice day!

    Well done! You seem to have warmed up!

    ……（其余阶段的验证输出）

实验工具

1. pobjdump：反汇编二进制炸弹程序，获得其中汇编指令供分析

    objdump -d bomb　输出bomb程序的反汇编结果

    objdump -d bomb > bomb.s　获得bomb程序的反汇编结果并保存于文本文件bomb.s中供分析

    objdump -t bomb 打印bomb程序的符号表，其中包含bomb中所有函数、全局变量的名称和存储地址

2. strings：显示二进制程序中的所有可打印字符串

3. gdb：强大的交互式程序调试工具（详细介绍可参看GDB文档和相关资料），可帮助从二进制可执行bomb程序中分析、找出触发bomb爆炸的条件，具体具有以下几方面功能：
    - 装载、启动被调试的程序
    - 让被调试的程序在指定的调试断点处中断执行，方便查看程序变量、寄存器、栈内容等运行现场数据
    - 动态改变程序的执行环境，如修改变量的值。跟踪二进制炸弹程序各阶段函数的运行，查看相关数据对象