作业2

2.1 一个C程序可以编译成目标文件或可执行文件。目标文件和可执行文件通常包含text、data、bss、rodata段，程序执行时也会用到堆（heap）和栈（stack）。
（1）请写一个C程序，使其包含data段和bss段，并在运行时包含堆的使用。请说明所写程序中哪些变量在data段、bss段和堆上。
（2）请了解readelf、objdump命令的使用，用这些命令查看（1）中所写程序的data和bss段，截图展示。
（3）请说明（1）中所写程序是否用到了栈。

提交内容：所写C程序、问题解答、截图等。

解答:
C 程序:

```c
#include<stdio.h>
#include<stdlib.h>

int global_var_data = 10;//全局变量存储在数据段
int global_var_bss;
int main(){
    static int local_static_var = 20;//初始化的静态局部变量存储在 data 段上
    static int local_var_bss;//未初始化的静态局部变量存储在 bss 段上
    int local_var=40;
    int *ptr;
    ptr =(int*)malloc(sizeof(int));//动态分配内存
    if(ptr==NULL)
    {
        printf("Memory not allocated.\n");
        exit(0);
    }
    *ptr = 30;//为堆上的数据赋值

    //打印变量
    printf("Value of local_var_bss is (bss sector): %d\n",local_var_bss);
    printf("Value of local_var is (stack): %d\n",local_var);
    printf("Value of local_static_var is (data
sector): %d\n",local_static_var);
    printf("Value of global_var is (data sector): %d\n",global_var_data);
    printf("Value of global_var is (bss): %d\n",global_var_bss);
    printf("Value of *ptr (heap sector)is : %d\n",*ptr);

    //释放内存
    free(ptr);
    return 0;
```

```
}
}
```

如上图所示的 C 程序，`.data` 段保存的是那些已经初始化了的全局静态变量和局部静态变量，`.bss` 段用于存储未初始化的全局和静态变量。函数内部，调用函数后，局部变量存储在函数的栈帧上。指针变量在运行时动态分配了堆上的内存来存储值。

运行结果：

```
Value of local_var_bss is (bss sector): 0
Value of local_var is (stack): 40
Value of local_static_var is (data sector): 20
Value of global_var is (data sector): 10
Value of global_var is (bss): 0
Value of *ptr (heap sector)is : 30
```

命令使用：

部分 elf 内容展示：

stu@stu:~/OSLab-RISC-V/HOMEWORK/Theory_course$ readelf -S homework2

```
There are 31 section headers, starting at offset 0x3aa8:
Section Headers:
  [Nr] Name              Type             Address           Offset
                                          .
                                          .
                                          .
[25] .data             PROGBITS         0000000000004000  00003000
       0000000000000018  0000000000000000  WA       0     0     8
[26] .bss              NOBITS           0000000000004018  00003018
       0000000000000010  0000000000000000  WA       0     0     4.
                                          .
                                          .
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  l (large), p (processor specific)
```

可以使用 `readelf -h elffile` 查看 ELF 文件头

```
readelf -h test
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x10e0
  Start of program headers:          64 (bytes into file)
```

```
  Start of section headers:          15056 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```

这部分内容其实就是操作系统研讨课实验一需要设置的信息等。

利用反汇编命令查看.bss 和.data 段(部分内容)

objdump -h homework2

```
homework2:     file format elf64-x86-64
[Nr] Name              Type             Address           Offset
       Size            EntSize          Flags  Link  Info  Align
                                                ...
24 .data               00000018  0000000000004000  0000000000004000  00003000  2**3
                       CONTENTS, ALLOC, LOAD, DATA
25 .bss                00000010  0000000000004018  0000000000004018  00003018  2**2
                       ALLOC
                                                ...
```

objdump -s homework2

```
Contents of section .data:
 4000 00000000 00000000 08400000 00000000  .........@......
 4010 0a000000 14000000                    ........
```

可以发现 .data 在编译为可执行文件后默认存在 00000000 00000000 08400000 00000000 四个四字节的数据，但目前我并不清楚是用于做什么，但是猜测可能是编译器在编译时定义的数据，这些数据即便未定义全局变量也会出现在.data 段中，其他俩个 0a000000 14000000 则分别对应 global_var_data 和 local_var,注意是小尾端译码。另外尽管段表中存在.bss 段的信息，但是 ELF 文件中并没有其内容，CONTENTS 表示该段在文件中存在。BSS 段没有 CONTENTS，表示它实际上在 ELF 文件中不存在内容。.bss 段存放的是未初始化的全局变量和局部静态变量。global _va_bss 和 local_var_bss 就是放在.bss 段, 其实更准确的说法是.bss 段为它们预留了空间。但是我们可以看到该段的大小有 16 个字节，通过符号表，我们可以看到符号表里只存了 local_var_bss 和 global_var_bss，以及其他一些预留数据。编译器可能会将全局的未初始化变量存放在目标文件.bss 段，有些则不存放，只是预留一个未定义的全局变量符号，等到最终链接成可执行文件的时候再在.bss 段分配空间。

符号表：objdump -t homework2

```
test:     file format elf64-x86-64

SYMBOL TABLE:(只留下.data 段和.bss 段)
0000000000004000 l     d  .data  0000000000000000              .data
0000000000004018 l     d  .bss   0000000000000000              .bss
```

```
0000000000004018 l     O .bss   0000000000000001              completed.8060
0000000000004014 l     O .data  0000000000000004              local_var.2834
000000000000401c l     O .bss   0000000000000004              local_var_bss.2835
0000000000004000   w       .data  0000000000000000              data_start
0000000000004020 g     O .bss   0000000000000004              global_var_bss
0000000000004018 g       .data  0000000000000000              _edata
0000000000004000 g       .data  0000000000000000              __data_start
0000000000004008 g     O .data  0000000000000000              .hidden
0000000000004028 g       .bss   0000000000000000              _end
0000000000004018 g       .bss   0000000000000000              __bss_start
0000000000004010 g     O .data  0000000000000004              global_var_data
0000000000004018 g     O .data  0000000000000000              .hidden __TMC_END__
```

C 程序通常会使用栈来存储函数调用的局部变量和函数调用信息，该程序定义自己的局部变量 local_var 变量，因此程序使用到了栈，

利用 gbd 调试工具，输入 info locals 可以查看栈帧中的局部变量如图：