实验 7 Windows 进程虚拟地址空间布局

目的：利用 objdump 分析可执行文件中代码段、数据段的分布，形成 Windows 下可执行文件结构的认识；利用 VMMap、Windbg、ProcessExplorer 等工具，分析进程虚拟地址空间布局。

步骤：

1、 使用熟悉的编辑器编辑下面的 C 文件。

virtual_mem.c

```
#include <stdio.h>
#include <string.h>
#include <memory.h>

/*函数 foo 的定义*/
int foo(int a, int b){
	return a+b;
}

/*函数 bar 的定义*/
int bar(int a, int b){
	return a*b;
}

/*定义全局变量，其中 x 和 z 被初始化，y 未被初始化*/
int x=10, y, z=20;

int main(int argc, char* argv[ ]){
/*局部变量*/
char buff[64];
int a=5, b, c=6;

/*动态内存分配*/
char* chp=(char*)malloc(16);

printf("(.text)address of \n\t foo=%p\n\t bar=%p\n\t main=%p\n", foo, bar, main);
printf("(.data initiated)address of \n\t x(inited)=%p\n\t z(inited)=%p\n",&x,&z);
printf("(.bss initiated)address of \n\t y(uninited)=%p\n",&y);
printf("(.stack) of \n\t argc=%p\n\t argv=%p\n\t argv[0]=%p\n", &argc, &argv, argv[0]);
printf("(Local variable) of \n\t buff[64]=%p\n", buff);
printf("(Local variable) of a \n\t a(inited)=%p\n\t b(uninited)=%p\n\t c(inited)=%p\n",&a,&b,&c);
printf("(.heap) of chp \n\t %p", chp);

/*让程序一直逗留在内存中以便观察进程镜像*/
```

```
while(1);
return 0;
}
```

2、 采用 VC 编译器编译此文件。打开命令行窗口，键入命令：

cl /Fd /Zi virtual_mem.c

生成 virtual_mem.obj，virtual_mem.exe，virtual_mem.pdb 等文件。

```
E:\操作系统教材和讲稿PPT\教材程序+案例程序\virtual-memory>cl /Fd /Zi virtual_mem.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

virtual_mem.c
Microsoft (R) Incremental Linker Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:virtual_mem.exe
/debug
virtual_mem.obj

E:\操作系统教材和讲稿PPT\教材程序+案例程序\virtual-memory>dir
 驱动器 E 中的卷是 新加卷
 卷的序列号是 C833-FC9D

 E:\操作系统教材和讲稿PPT\教材程序+案例程序\virtual-memory 的目录

2018/11/19  21:06    <DIR>          .
2018/11/19  21:06    <DIR>          ..
2018/11/19  21:17            45,056 vc60.pdb
2018/11/19  21:02               989 virtual_mem.c
2018/11/19  21:17            65,626 virtual_mem.exe
2018/11/19  21:17           117,320 virtual_mem.ilk
2018/11/19  21:06             1,586 virtual_mem.o
2018/11/19  21:17             2,781 virtual_mem.obj
2018/11/19  21:17           173,056 virtual_mem.pdb
               7 个文件        406,414 字节
               2 个目录 91,161,075,712 可用字节
```

3、 采用 objdump 工具观察可执行文件的结构。在命令行窗口键入命令：

objdump -h virtual_mem.exe

```
E:\操作系统教材和讲稿PPT\教材程序+案例程序\virtual-memory>objdump -h virtual_mem.exe

virtual_mem.exe:     file format pei-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00007ddf  00401000  00401000  00001000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rdata        00000601  00409000  00409000  00009000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .data         00004000  0040a000  0040a000  0000a000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .idata        0000067a  00410000  00410000  0000e000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .reloc        0000077b  00411000  00411000  0000f000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
```

从中可以看出：

- 可执行文件的格式是 PE 格式，即 Windows 上的可执行文件格式。Linux 上可执行文件

为 ELF 格式。

- 代码段.text 大小为 0x00007ddf 字节，在可执行文件中的偏移量是 0x00001000，而且是 READONLY（只读）的。在可执行文件中代码段已经分页了，页面大小为 4KB，即 0x1000 字节。偏移量 0x00001000 恰好是 1 号页面的起始地址（页面号是从 0 号开始编址）。

  由于.text 段大小为 0x00007ddf，需要 8 个页面来保存，因此它在可执行文件中的地址范围是：0x00001000~0x00008FFF。
- 只读数据段.rdata 大小为 0x00000601，不到一个页面大小，因此划分在一个页面中。它在可执行文件中的偏移量是 0x00009000，因此其地址范围是：0x00009000~0x00009FFF。
- 可读可写数据段.data 大小为 0x00004000，恰好占 4 个页面，它在可执行文件中的偏移量是 0x0000a000，因此其地址范围是 0x0000a000~0x0000dFFF。

4、观察虚拟地址空间分布。在上面第 3 步中，各个段除了在可执行文件中按照页面分布外，实际上，已经被映射进进程的虚拟地址空间。

- 代码段.text 的虚拟地址偏移量为 0x00401000，即 VMA 列的值。这个起始地址恰好是 0x00401 号页面的起始地址。
- 只读数据段.rdata 的虚拟地址偏移量为 0x00409000，恰好是 0x00409 号页面的起始地址。
- 可读可写数据段.data 的虚拟地址偏移量为 0x0040a000，恰好是 0x0040a 号页面的起始地址。

我们用下表建立各个段在可执行文件中的地址分布与在虚拟地址空间中的地址分布之间的关系，这一关系就是虚拟地址空间 VM 到磁盘文件空间 File 之间的 Allocate 映射。

表 1　Allocate 映射：VM-> File

逻辑段（section）	文件中的分布	虚拟地址空间中的分布
.text	0x00001000~0x00008FFF 占 8 个页面	0x00401000~_____
.rdata	0x00009000~0x00009FFF 占 1 个页面	0x00409000~_____
.data	0x0000a000~0x0000dFFF 占 4 个页面	0x0040a000~_____

5、运行程序 virtual_mem.exe。

```
E:\操作系统教材和讲稿PPT\教材程序+案例程序\virtual-memory>virtual_mem
(.text)address of
        foo=0040100A
        bar=00401005
        main=0040100F
(.data initiated)address of
        x(inited)=0040AA30
        z(inited)=0040AA34
(.bss initiated)address of
        y(uninited)=0040DF40
(.stack) of
        argc=0019FF48
        argv=0019FF4C
        argv[0]=00890EF0
(Local variable) of
        buff[64]=0019FEF0
(Local variable) of a
        a(inited)=0019FF3C
        b(uninited)=0019FF38
        c(inited)=0019FF34
(.heap) of chp
        00890D98
```

得到各个函数以及变量的地址。这些地址能够指示每个段的大致分布。

问答题：

1、 填写表 1 中空白，画出虚拟地址空间到磁盘文件空间的映射关系。

2、 .text 段的虚拟地址为 0x00401000，而函数 main 的地址是 0x0040100F，为什么.text 段的入口和 main 的入口不一致，这能够说明什么问题？

3、 从程序运行结果分析，x 和 z 这两个变量的虚拟内存位置是否相邻？整型数占几个字节？变量 y 的内存与 x、z 的内存分布为什布不相邻？

4、 如果用 gcc 编译 virtual_mem.c，得到的可执行文件格式与 cl 编译结果一致吗？