观察内存映射

实验原理

- **程序进程中的分区**：
- **BSS段**：BSS段（bss segment）通常是指用来存放程序中**未初始化的全局变量**的一块内存区域。
  数据段：数据段（data segment）通常是指用来存放程序中**已初始化的全局变量**的一块内存区域。
  代码段：代码段（code segment/text segment）通常是指用来存放程序**执行代码**的一块内存区域。这部分区域的大小在程序运行前就已经确定，并且内存区域通常属于只读, 某些架构也允许代码段为可写，即允许修改程序。在代码段中，也有可能包含一些只读的常数变量，例如字符串常量等。
  堆(heap)：堆是用于存放进程运行中**被动态分配的**内存段，它的大小并不固定，可动态扩张或缩减。当进程调用malloc等函数分配内存时，新分配的内存就被动态添加到堆上（堆被扩张）；当利用free等函数释放内存时，被释放的内存从堆中被剔除（堆被缩减）。
  栈(stack)：栈又称堆栈，用户存放程序**临时创建的局部变量**。在函数被调用时，其参数也会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。由于栈的后进先出特点，所以栈特别方便用来保存/恢复调用现场。

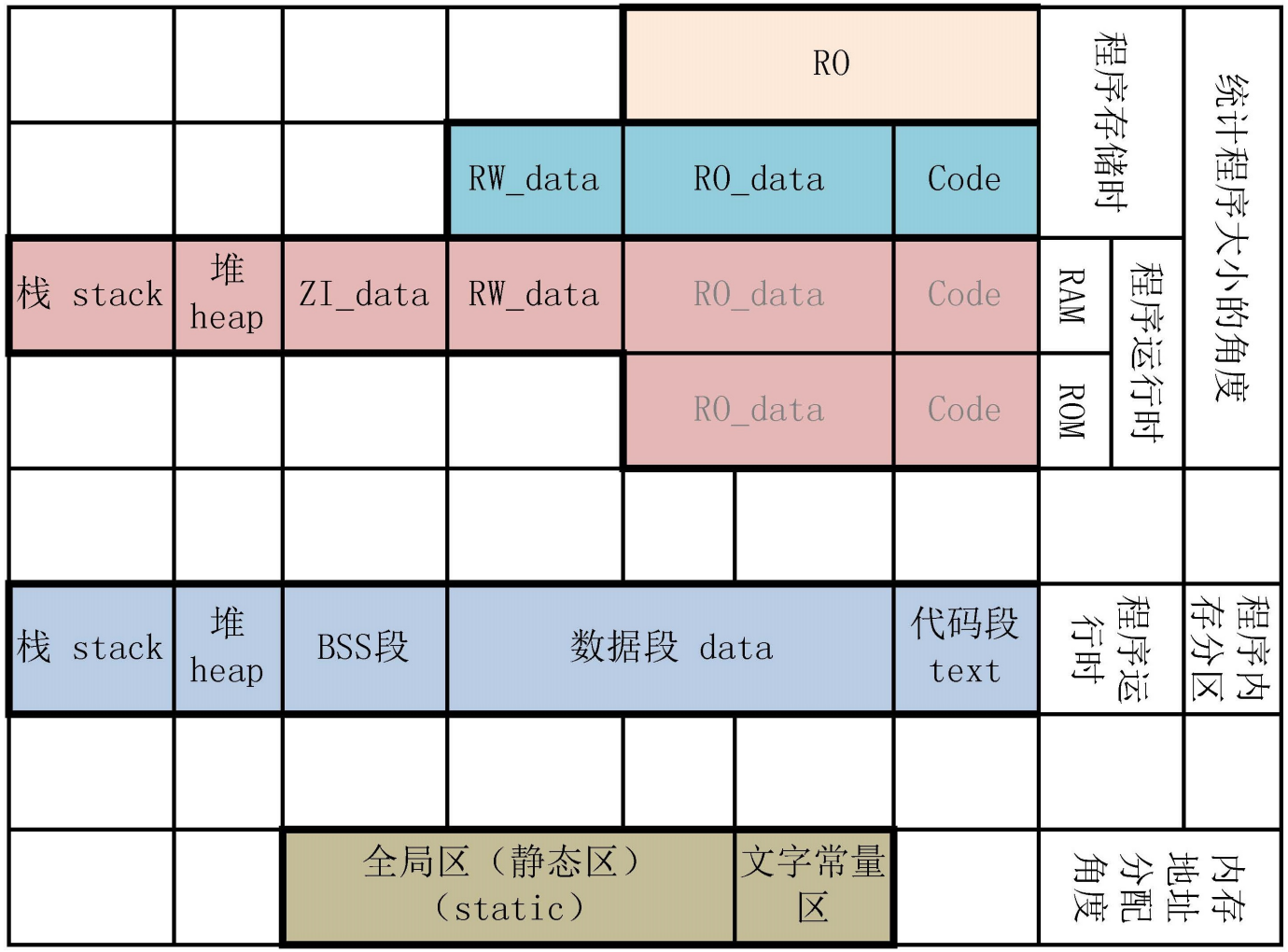
实验原理

- **程序文件中的分区**
- **Code**：即代码域，它指的是编译器生成的机器指令。
  RO_data：ReadOnly data，即**只读数据域**，它指程序中用到的只读数据，全局变量，例如C语言中const关键字定义的全局变量就是典型的RO-data。
  RW_data：ReadWrite data，即可读写数据域，它指**初始化为“非0值”的可读写数据**，程序刚运行时，这些数据具有非0的初始值，且运行的时候它们会常驻在RAM区，因而应用程序可以修改其内容。例如全局变量或者静态变量，且定义时赋予“非0值”给该变量进行初始化。
  ZI_data：ZeroInitialied data，即0初始化数据，它指**初始化为“0值”的可读写数据域**，它与RW_data的区别是程序刚运行时这些数据初始值全都为0，而后续运行过程与RW-data的性质一样，它们也常驻在RAM区，因而应用程序可以更改其内容。包括未初始化的全局变量，和初始化为0的全局变量。
  RO：只读区域，包括RO_data和code。

实验原理

当程序存储在ROM中时，所占用的大小为Code + RO_data + RW_data 。
当程序执行时， RW_data和 ZI_data在RAM中， RO_data和code视cpu架构（51、arm、x86）不同处于ROM或者RAM中。其中ZI_data对应了BSS段，RW_data对应数据段，code对应代码段，RO_data对应数据段。

实验原理


RO
RW_data
RO_data
Code
栈 stack
堆 heap
ZI_data
RW_data
RO_data
Code
RO_data
Code
栈 stack
堆 heap
BSS段
数据段 data
代码段 text
全局区（静态区）（static）
文字常量区
程序存储时
程序运行时
RAM
ROM
统计程序大小的角度
程序运行时
程序内存分区
内存地址分配角度

实验目的

观察未初始化/初始化的全局变量、动态局部变量(heap)、临时局部变量(stack)的内存映射

```
int globalvar1;

int globalvar2 = 3;

void mylocalfoo()

{

  int functionvar;

  printf("variable functionvar \t location: 0x%lx\n", &functionvar);

}

int main()

{

  void *localvar1 = (void *)malloc(2048);
```

实验流程

1. 编译链接singlefoo.c 为shared library

>> gcc -o liblkpsinglefoo.so -O2 -fPIC -shared lkpsinglefoo.c

2.拷贝到动态库到默认动态库路径

>> sudo cp liblkpsinglefoo.so /usr/lib/

3. 编译lkpmem.c

>> gcc lkpmem.c liblkpsinglefoo.so -o lkpmem

4. 运行测试程序

>> ./lkpmem

5.查看该测试程序的内存映射

>> ps aux | grep lkpmem

>> cat /proc/<pid>/maps

实现效果

```
amos@ubuntu:~/Desktop/5$ ./lkpmem
variable globalvar1       location: 0x5555d810f018
variable globalvar2       location: 0x5555d810f010
variable localvar1        location: 0x7ffe739fa950
variable libvar           location: 0x7ffe739fa934
variable functionvar      location: 0x7ffe739fa934
```

```
amos@ubuntu:~/Desktop/5$ cat /proc/35510/maps
5555d7f0e000-5555d7f0f000 r-xp 00000000 08:01 1843577                    /home/amos/Desktop/5/lkpmem
5555d810e000-5555d810f000 r--p 00000000 08:01 1843577                    /home/amos/Desktop/5/lkpmem
5555d810f000-5555d8110000 rw-p 00001000 08:01 1843577                    /home/amos/Desktop/5/lkpmem
5555d88bf000-5555d88e0000 rw-p 00000000 00:00 0                          [heap]
7fc8e5708000-7fc8e58ef000 r-xp 00000000 08:01 1840977                    /lib/x86_64-linux-gnu/libc-2.27.so
7fc8e58ef000-7fc8e5aef000 ---p 001e7000 08:01 1840977                    /lib/x86_64-linux-gnu/libc-2.27.so
7fc8e5aef000-7fc8e5af3000 r--p 001e7000 08:01 1840977                    /lib/x86_64-linux-gnu/libc-2.27.so
7fc8e5af3000-7fc8e5af5000 rw-p 001eb000 08:01 1840977                    /lib/x86_64-linux-gnu/libc-2.27.so
7fc8e5af5000-7fc8e5af9000 rw-p 00000000 00:00 0
7fc8e5af9000-7fc8e5afa000 r-xp 00000000 08:01 1051836                    /usr/lib/liblkpsinglefoo.so
7fc8e5afa000-7fc8e5cf9000 ---p 00001000 08:01 1051836                    /usr/lib/liblkpsinglefoo.so
7fc8e5cf9000-7fc8e5cfa000 r--p 00000000 08:01 1051836                    /usr/lib/liblkpsinglefoo.so
7fc8e5cfa000-7fc8e5cfb000 rw-p 00001000 08:01 1051836                    /usr/lib/liblkpsinglefoo.so
7fc8e5cfb000-7fc8e5d22000 r-xp 00000000 08:01 1840949                    /lib/x86_64-linux-gnu/ld-2.27.so
7fc8e5f05000-7fc8e5f0a000 rw-p 00000000 00:00 0
7fc8e5f22000-7fc8e5f23000 r--p 00027000 08:01 1840949                    /lib/x86_64-linux-gnu/ld-2.27.so
7fc8e5f23000-7fc8e5f24000 rw-p 00028000 08:01 1840949                    /lib/x86_64-linux-gnu/ld-2.27.so
7fc8e5f24000-7fc8e5f25000 rw-p 00000000 00:00 0
7ffe739db000-7ffe739fc000 rw-p 00000000 00:00 0                          [stack]
7ffe739fc000-7ffe739ff000 r--p 00000000 00:00 0                          [vvar]
7ffe739ff000-7ffe73a00000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

maps说明

内核每进程的vm_area_struct项	/proc/pid/maps中的项	含义
vm_start	“-”前一列，如00377000	此段虚拟地址空间起始地址
vm_end	“-”后一列，如00390000	此段虚拟地址空间结束地址
vm_flags	第三列，如r-xp	此段虚拟地址空间的属性。每种属性用一个字段表示，r表示可读，w表示可写，x表示可执行，p和s共用一个字段，互斥关系，p表示私有段，s表示共享段，如果没有相应权限，则用’ -’ 代替
vm_pgoff	第四列，如00000000	对有名映射，表示此段虚拟内存起始地址在文件中以页为单位的偏移。对匿名映射，它等于0或者vm_start/PAGE_SIZE
vm_file->f_dentry->d_inode->i_sb->s_dev	第五列，如fd:00	映射文件所属设备号。对匿名映射来说，因为没有文件在磁盘上，所以没有设备号，始终为00:00。对有名映射来说，是映射的文件所在设备的设备号
vm_file->f_dentry->d_inode->i_ino	第六列，如9176473	映射文件所属节点号。对匿名映射来说，因为没有文件在磁盘上，所以没有节点号，始终为0。对有名映射来说，是映射的文件的节点号
	第七列，如/lib/ld-2.5.so	对有名来说，是映射的文件名。对匿名映射来说，是此段虚拟内存在进程中的角色。[stack]表示在进程中作为栈使用，[heap]表示堆。其余情况则无显示