操作系统 作业 1

张远航 2015K8009929045

2017 年 9 月 10 日

编译运行 addr_space.c 的结果（注：每次运行时， ldata 和 ddata 的地址都会发生变化）：

```
yuan-hang@caszhang:~/桌面/os/hw1$ gcc -w -o addr_space addr_space.c
yuan-hang@caszhang:~/桌面/os/hw1$ ./addr_space
gdata:6010A0
bdata:601060
ldata:7ffdb0d08590
ddata:1a72010
```

按地址空间的分配原则，gdata 属于未初始化的全局数据，应位于数据段的.bss 区域；数组首指针 bdata 和字符串首指针 myname 是已初始化的非零数据（注： myname 随存放的的常量一同初始化，指向的字符串常量位于 .rodata 段），应位于数据段的.data 区域；ldata 是动态分配空间的局部变量，位于高地址的堆栈段；ddata 是运行时动态分配的内存空间，应位于数据段的堆区域。

objdump 的结果证实， gdata 位于 .bss 段，而 bdata 和 myname 位于.data 段：

```
yuan-hang@caszhang:~/桌面/os/hw0$ objdump -t addr_space | grep gdata
00000000006010a0 g     O .bss   0000000000000080              gdata
yuan-hang@caszhang:~/桌面/os/hw0$ objdump -t addr_space | grep bdata
0000000000601060 g     O .data  0000000000000010              bdata
yuan-hang@caszhang:~/桌面/os/hw0$ objdump -t addr_space | grep myname
0000000000601050 g     O .data  0000000000000008              myname
```

还可以进一步找到 bdata 和 myname 中存放的常量：

```
yuan-hang@caszhang:~/桌面/os/hw1$ objdump -s addr_space
....
Contents of section .data:
601040 00000000 00000000 00000000 00000000  ................
601050 38074000 00000000 00000000 00000000  8.@.............
601060 01020304 00000000 00000000 00000000  ................
....
```

可以看到以 bdata 为首地址存放的 1、2、3、4 四个数。myname 指向的地址为00400738，用 objdump 可以看到存放在 .rodata 段的字符串常量：

```
Contents of section .rodata:
400730 01000200 00000000 42616f20 59756e67  ........Bao Yung
400740 616e6700 00000000 67646174 613a256c  ang.....gdata:%l
400750 6c580a62 64617461 3a256c6c 580a6c64  lX.bdata:%llX.ld
400760 6174613a 256c6c78 0a646461 74613a25  ata:%llx.ddata:%
```

```
400770 6c6c780a 00                              llx..
```

用 gcc -w -gdwarf-2 -o addr_space addr_space.c 命令编译带 DWARF 调试信息的二进制文件后后，执行 objdump：

```
yuan—hang@caszhang:~/桌面/os/hw1$ objdump —S addr_space
addr_space:     文件格式 elf64—x86—64
....
0000000000400626 <main>:
#include <stdlib.h>

char *myname="Bao Yungang";
char gdata[128];
char bdata[16] = {1,2,3,4};
main() {
        400626: 55                      push   %rbp
        400627: 48 89 e5                mov    %rsp,%rbp
        40062a: 48 81 ec a0 00 00 00    sub    $0xa0,%rsp
        400631: 64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
        400638: 00 00
        40063a: 48 89 45 f8             mov    %rax,—0x8(%rbp)
        40063e: 31 c0                   xor    %eax,%eax
        char * ldata[16];
        char * ddata;

        ddata = malloc(16);
        400640: bf 10 00 00 00          mov    $0x10,%edi
        400645: e8 c6 fe ff ff          callq  400510 <malloc@plt>
        40064a: 48 89 85 68 ff ff ff    mov    %rax,—0x98(%rbp)
        printf("gdata:%llX\nbdata:%llX\nldata:%llx\nddata:%llx\n",
        400651: 48 8b 95 68 ff ff ff    mov    —0x98(%rbp),%rdx
        400658: 48 8d 85 70 ff ff ff    lea    —0x90(%rbp),%rax
        40065f: 49 89 d0                mov    %rdx,%r8
        400662: 48 89 c1                mov    %rax,%rcx
        400665: ba 60 10 60 00          mov    $0x601060,%edx
        40066a: be a0 10 60 00          mov    $0x6010a0,%esi
        40066f: bf 48 07 40 00          mov    $0x400748,%edi
        400674: b8 00 00 00 00          mov    $0x0,%eax
        400679: e8 72 fe ff ff          callq  4004f0 <printf@plt>
        gdata,bdata,ldata,ddata);
        free(ddata);
        40067e: 48 8b 85 68 ff ff ff    mov    —0x98(%rbp),%rax
        400685: 48 89 c7                mov    %rax,%rdi
        400688: e8 43 fe ff ff          callq  4004d0 <free@plt>
        40068d: b8 00 00 00 00          mov    $0x0,%eax
}
```

```
400692: 48 8b 4d f8              mov    —0x8(%rbp),%rcx
400696: 64 48 33 0c 25 28 00     xor    %fs:0x28,%rcx
40069d: 00 00
40069f: 74 05                    je     4006a6 <main+0x80>
4006a1: e8 3a fe ff ff           callq  4004e0 <__stack_chk_fail@plt>
4006a6: c9                       leaveq
4006a7: c3                       retq
4006a8: 0f 1f 84 00 00 00 00     nopl   0x0(%rax,%rax,1)
4006af: 00
....
```

其中，进入 `main` 函数时：

```
main() {
400626: 55                       push   %rbp
400627: 48 89 e5                 mov    %rsp,%rbp
40062a: 48 81 ec a0 00 00 00     sub    $0xa0,%rsp
```

在 64 位系统中，一个指针变量占 8 个字节。这三行代码将栈顶指针减小 `0xa0`，即共为 `ldata` 数组和 `ddata` 分配了 20×8 字节的栈空间。

调用 `malloc` 为 `ddata` 分配内存时，新分配的内存被动态添加到堆上；调用 `free` 释放内存时，被释放内存被从堆中移除。