1. 有程序段如下:

|  |
| --- |
| **int** foo() {  **char** str1[20], \*str2;  str2 = (char\*)malloc(20 \* sizeof char);  free(str2);  } |

下列说法中正确的是\_\_\_\_**C**\_\_\_.

A.str1 和 str2 指向的内存都是分配在栈空间内的

B.str1 和 str2 指向的内存都是分配在堆空间内的

C.str1 指向的内存是分配在栈空间内的,str2 指向的内存是分配在堆空间内的

D.str1 指向的内存是分配在堆空间内的,str2 指向的内存是分配在栈空间内的

1. 在某一 64 位体系结构中，每页的大小为 4KB，采用的是三级页表，每张页表占据 1 页，页表项长度为 8 字节。则虚拟地址的位数为\_\_9\_\_Bit。如果要映射满 64 位的虚拟地址空间，可通过增加页表级数来解决，那么至少要增加到\_\_\_\_\_级页表。这个体系结构支持多种页大小，最小的三个页大小分别是 4KB、\_\_\_\_\_MB、\_\_\_\_\_GB

**【答】VPO 长度为 12 位。4KB/32=512 个条目每页表。因此 VPN 长度均为 9 位，虚拟地址长度为 12+9+9+9=39 位。由于 12+9\*5=57<64，12+9\*6=66>=64，因此需要采用6级页表才能映射满。 2 1**

1. 动态管理器分配策略中,最适合“最佳适配算法”的空白区组织方式是:**B**

A.按大小递减顺序排列

B.按大小递增顺序排列

C.按地址由小到大排列

D.按地址由大到小排列

1. 有如下程序内容：

|  |
| --- |
| #include "csapp.h"  const char\* hello = "No use\n";  char\* bye = NULL;  int fd1 = 1;  int fd2;  int main() {  fd1 = open("hello.txt", O\_RDWR);  fd2 = open("bye.txt", O\_RDWR);  hello = mmap(NULL, 16, PROT\_READ, MAP\_SHARED, fd1, 0);  bye = mmap(NULL, 16, PROT\_READ | PROT\_WRITE,  MAP\_SHARED, fd2, 0);  for (int i = 0; i < 8; i++)  bye[i] = toupper(hello[i]);  /\*\*\*\*\*\*\*\*\*\*\*\*A\*\*\*\*\*\*\*\*\*\*\*\*/  munmap(hello, 16);  munmap(bye, 16);  return 0;  } |

1. 将以下符号归属到各个section中:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| symbol | text | data | bss | 不是符号 |
| main | **√** |  |  |  |
| hello |  | **√** |  |  |
| bye |  |  | **√** |  |
| fd1 |  | **√** |  |  |
| fd2 |  |  | **√** |  |
| i |  |  |  | **√** |

1. 代码运行到 A 处的时候，/proc/2333/maps 中的内容如下：

|  |  |  |
| --- | --- | --- |
| ADDRESS | PERM | PATH |
| 555555555000-555555556000 | (1) | /home/eugen/vm/vm |
| 555555556000-555555557000 | r--p | /home/eugen/vm/vm |
| 555555558000-555555559000 | rw-p | /home/eugen/vm/vm |
| 7ffff7dea000-7ffff7f62000 | r-xp | /usr/lib/x86\_64-linux-gnu/libc-2.31.so |
| 7ffff7f62000-7ffff7fb4000 | r--p | /usr/lib/x86\_64-linux-gnu/libc-2.31.so |
| 7ffff7fb4000-7ffff7fb6000 | rw-p | /usr/lib/x86\_64-linux-gnu/libc-2.31.so |
| 7ffff7fc9000-7ffff7fca000 | (2) | /home/eugen/vm/bye.txt |
| 7ffff7fca000-7ffff7fce000 | r--p | [vvar] |
| 7ffff7fce000-7ffff7fcf000 | r-xp | [vdso] |
| 7ffff7fcf000-7ffff7fd0000 | r--p | /usr/lib/x86\_64-linux-gnu/(4)-2.31.so |
| 7ffff7fd0000-7ffff7ff3000 | r-xp | /usr/lib/x86\_64-linux-gnu/(4)-2.31.so |
| 7ffff7ff3000-7ffff7ffb000 | r--p | /usr/lib/x86\_64-linux-gnu/(4)-2.31.so |
| 7ffff7ffb000-7ffff7ffc000 | (3) | /home/eugen/vm/hello.txt |
| 7ffff7ffc000-7ffff7ffd000 | r--p | /usr/lib/x86\_64-linux-gnu/(4)-2.31.so |
| 7ffff7ffd000-7ffff7ffe000 | rw-p | /usr/lib/x86\_64-linux-gnu/(4)-2.31.so |
| 7ffffffde000-7ffffffff000 | rw-p | [(5)] |

PERM 有四位。前三位是 r=readable，w=writeable，x=executable，如果是-表示

没有这一权限。第四位是 s=shared，p=private，表示映射是共享的还是私有的。填写

(1)-(5)的内容：

**1: r-xp**

**2: rw-s**

**3: r--s**

**4: ld**

**5: stack**

7ffff7ffd000 对应的页在页表中被标为了只读。对该页进行写操作会导致Page Fault吗？ 会导致该进程收到SIGSEGV信号吗？

**【答】会发生Page fault,但是不会收到SIGSEGV信号。在内核的映射表中标出它是可写的，页表中之所以标记为只读，是因为 COW(copy on write) 还未发生。**