

# E03 Unix V6++ 进程图像、地址映射 和 内存保护

同济大学计算机系操作系统习题参考答案

姓名

学号

一、Unix V6++ 编译器的编译规则是：代码段起始虚地址 0x00401000。随后，依次存放数据段、只读数据段、bss 段，4K 对齐。栈底 0x00800000，初始栈 4096 字节。  
matrixSquare 程序，源代码如下，可执行文件头 2K 字节。

```
=====
#define M 32          // 注意，M 是编译器用的常数，不是可执行文件的只读数据
int matrixOriginal[M][M];
int matrixDes [M][M];
int main()
{
    int i , j;

    ...;
    for (i=0;i<M;i++)
L:      for (j=0;j<M;j++)
            produce(i , j);
    ...;
}
void produce ( int row, int column )
{
    int i;
    for (i=0;i<M;i++)
L1:      matrixDes[row][column] += matrixOriginal[row][i] * matrixOriginal[i][column];
}
=====
```

请回答下列问题：

(1) 如上述程序汇编后形成的机器语言指令为 1K，整型数占 4 个字节，请绘制上述代码汇编后形成的可执行文件的结构。

答：所有逻辑段的长度：

代码段：4K 字节

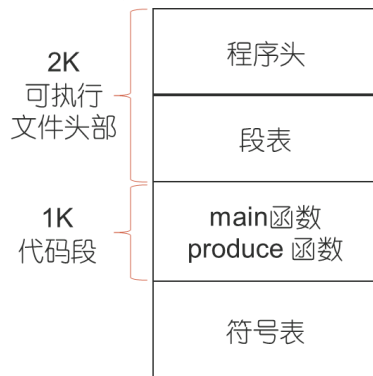
数据段：无

只读数据段：无

bss 段：(32\*32\*4) \* 2 = 8K 字节

堆栈段：4K 字节

## 1、可执行文件结构

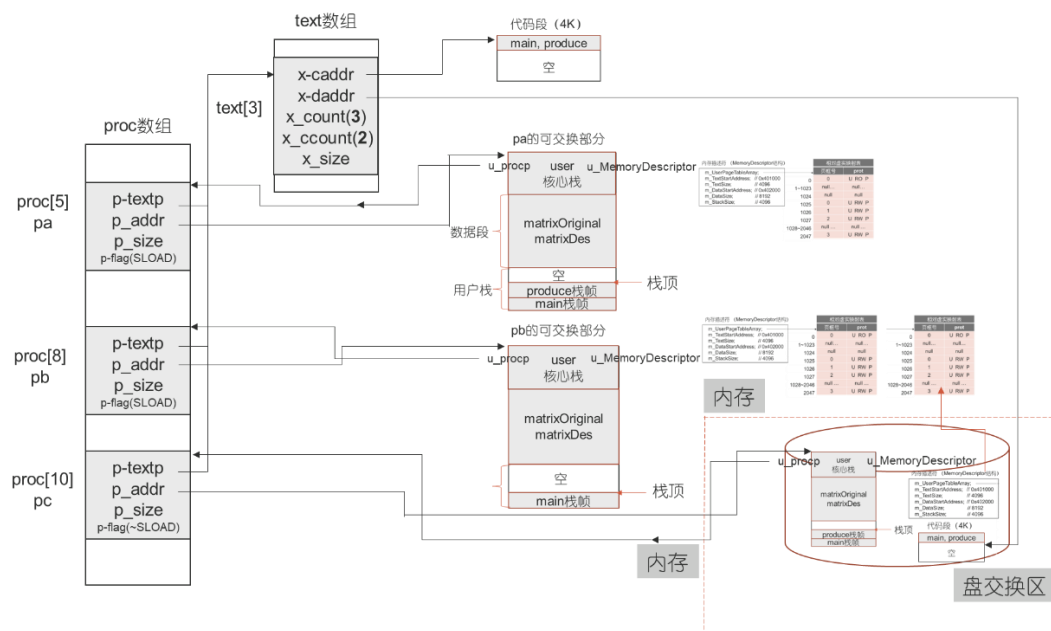


magic number = "PE\0\0"  
 section number = 3  
 entry point = &main1

段号	段名	读写属性	内存起始地址	段长	文件起始偏移量	文件中的长度
0	.code	RO, X	0x401000	4096	0x800	1024
1	.bss	RW, ~X	0x402000	8192	0	0
2	.stack	RW, ~X	null	4096	0	0

2、分别创建进程 **pa**, **pb**, **pc** 执行上述可执行文件，其中，**pa**, **pb** 在内存，**pc** 在盘交换区上，请绘制所有进程的图像，包括内存描述符。这 3 个进程相对虚实地址映射表是相同的，对吗？画出来相对虚实地址映射表。

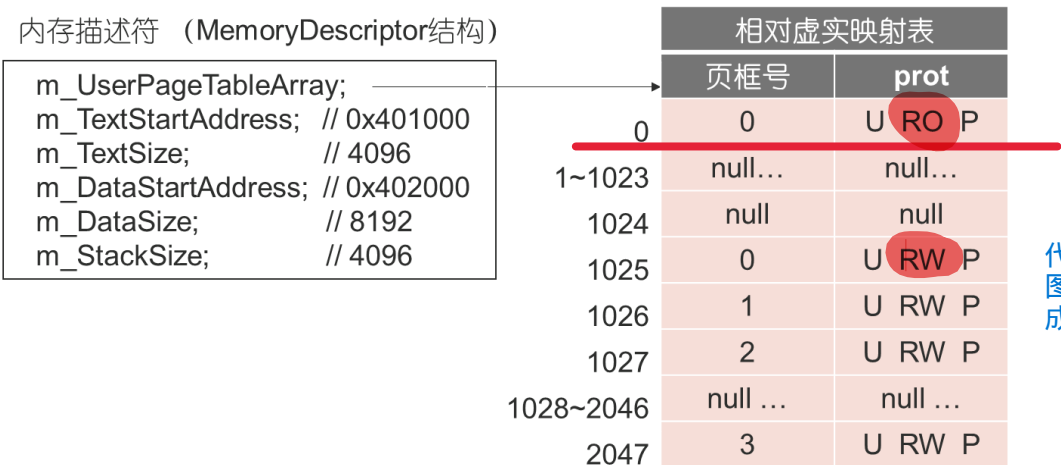
答：pa、pb、pc 进程的图像



图中，**pa**、**pc** 正在执行 **produce** 函数。**pb** 正在执行 **main** 函数。

内存描述符 **MemoryDescriptor** 的细节：

昨天累了，糊涂了，不该这里注释。相对表里，不该出现绝对的物理地址。属性，RO，RW没太大关系。建立映射关系的时候会覆盖。看MapToPageTable最后的3条语句。这个PTE不要求掌握。



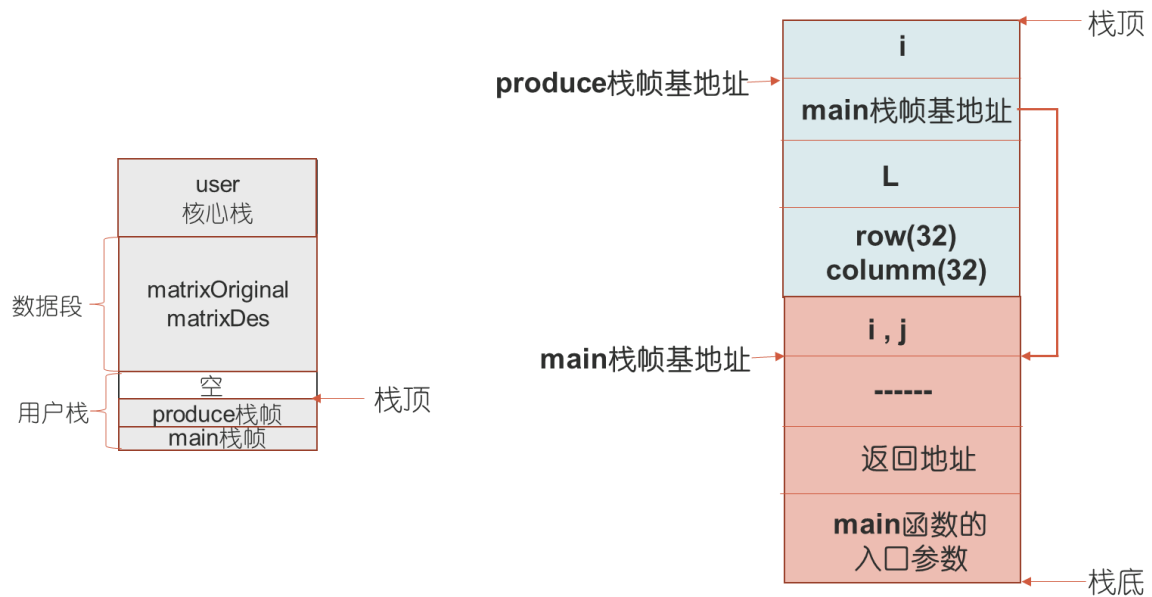
进程内存描述符的初始值（包括相对表）由可执行文件的段表内容决定。所以刚开始执行时，执行同一个程序的所有进程内存描述符是相同的。本题，程序没有 malloc 操作，子程序调用情况也较简单，不会出现栈溢出的情况，因此 3 个进程运行期间，内存描述符的值（包括相对表）完全相同。

注意：每个进程的内存描述符是私有的，执行同一个程序的多个进程不能共享。原因是：进程运行时，如果堆扩展 或 栈扩展，内存描述符就变了。执行同一个应用程序的所有进程彼此独立，会有执行快慢和走不同分支的情况。所以，运行一段时间后，执行同一个程序的多个进程，数据段和栈的长度，尤其是相对表会不一样。每个进程必需有自己的内存描述符。

3、T 时刻，现运行进程 pa，p\_addr == 0x500000，x\_caddr == 0x600000。画系统页表。



4、当 pa 执行到 produce 函数的 for 语句时，请绘制该进程的用户栈的构成（图中需标出 main 函数中的 i 和 j，produce 函数中的 row 和 column，i 和 j 的位置）。命令行参数环境变量的占位，考试不要求。



5、计算矩阵元素 `matrixDes [1][2]` 的虚地址。

答：进程数据段起始地址是 0x402000。

矩阵 `matrixOriginal` 尺寸是：32\*32\*4=4096

矩阵 `matrixDes` 的起始地址是：0x403000

`matrixDes [1][2]`，相较矩阵起始地址，偏移  $32+2=34$  个元素，共计 10001000（2 进制）个字节。换算成 16 进制，0x88。所以，该矩阵元素的虚地址是 0x403088。

6、MMU 实施的内存保护和地址映射操作

- 分割虚地址 0x403088，得页表号 1，页号 3，页内偏移量 0x88。
- 用页表号 1 做下标查页目录，1#页表在 0x203#物理页框。
- 用页号 3 做下标查 1#页表（也就是 0x203#物理页框），得映射 `matrixDes [1][2]` 元素需要的 PTE。蓝框框出的页表项。
  - `base = 0x502`，`matrixDes [1][2]` 元素在物理页框 0x502。
  - `prot = U R W P`。U，这是用户页面，允许应用程序访问。RW，这是数据页面，允许写入。P，页面内存中存在。
- 内存保护操作：
 

当前，pa 进程正在执行应用程序，CPU 用户态运行。内存访问，写操作。

权限检测通过，允许访问目标页面（也就是 0x502 物理页框）。
- 地址映射：
 

生成物理地址：0x502088。