2152118 史君宝 exec 和堆空间管理 订正

第一题：

一、这个程序的输出是什么?

```
#include <stdio.h>
#include <stdlib.h>

int main1( )   // tryExec.c
{
   char* argv[4];
   argv[0] = "showCmdParam";
   argv[1] = "arg1";
   argv[2] = "arg2";
   argv[3] = 0;

   if ( fork( ) ==0 )
         ;
   else{
      execv(argv[0] , argv) ;
      printf("Over!\n");
   }
   exit(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>

int main1(int argc, char *argv[])   // showCmdParam.c
{
   int i;

   printf("The command parameter of showCmdParam\n");

   for(i = 0; i < argc; i++)
      printf("argv[%d]:\t%s\n", i, argv[i]);

   exit(0);
}
```

自己的答案：

解:
我们通读上面的程序，可以知道程序的输出应该是:
输出如下：
The command parameter of showCmdParam (这里是换行)
argv[0]: showCmdParam (这里是换行)
argv[1]: arg1 (这里是换行)
argv[2]: arg2 (这里是换行)
Over! (这里是换行)

参考答案：

```
The command parameter of showCmdParam
argv[0]:  showCmdParam
argv[1]:  arg1
argv[2]:  arg2
```

注意（1）输出没有"over!" 因为，execv 不是普通的子程序调用，装载新程序 showCmdParam 后，原先用户空间中的 tryExec 程序没了，进程执行不到 printf("Over!\n")这条语句。（2）本题，是父进程在转换进程图像。

第二题：

二、现运行进程 PA，1 页代码，1 页数据，没有只读数据 和 bss，1 页堆栈。代码段起始 0x401000。进程依次执行下列动态内存分配释放操作。

```
char *p1= malloc(4);
char *p2= malloc(4);
char *p3= malloc(32);
free(p2);
char *p4= malloc(8);
free(p1);
char *p5= malloc(8);
```

(1) 指针 p1 的值是多少 ？
(2) 指针 p2 的值是多少 ？
(3) 指针 p3 的值是多少 ？
(4) 指针 p4 的值是多少 ？
(5) 指针 p5 的值是多少 ？ (6) 情景分析
(7) 画最终的堆结构图

自己的答案：

```
char *p1= malloc(4);
char *p2= malloc(4);
char *p3= malloc(32);
free(p2);
char *p4= malloc(8);
free(p1);
char *p5= malloc(8);
```

(1) 指针 p1 的值是多少 ？
指针 p1 的值是 0X403010
(2) 指针 p2 的值是多少 ？
指针 p1 的值是 0X403020
(3) 指针 p3 的值是多少 ？
指针 p1 的值是 0X403030
(4) 指针 p4 的值是多少 ？
指针 p1 的值是 0X403020
(5) 指针 p5 的值是多少 ？

指针 p1 的值是 0X403010

(6) 情景分析

解：我们对上面的过程有下面的情景分析：
首先，代码段的起址地址为 0x401000，长度为 1 页。
数据段的起址地址为 0x402000，长度为 1 页。
堆栈段的起址地址为 0x403000，
之后我们为数据段追加 12K 的数据，即 malloc_head=0x403000，malloc_end=0x406000，其中 malloc_head 的位置进行初始化时会写入一个 flist 结构，它的大小 8 字节。我们考察 p1 的内存片的构成，其中 flist 的大小为 8B，need 的大小是 4B，则大小应为 8+4=12B，对齐之后的大小为 16B；同时 p1 指向 malloc 的返回值。这个返回值指向的是 flist 的尾部，并不是内存片；因此 p1 指向的地址为 0x403010。
之后我们向内存片 1 的后面追加 flist 结构和 8B，此时 p2 指向的地址是 0x403020，p3 指向的地址是 0x403030。这时候内存片 2 会被释放。
内存片 1、3 之间大小为 8+8=16B，满足 p4，故 p4 指向的地址为 0x403020。
之后 malloc_head 结构的 flist 与内存片 4 之间的区域大小为 8+8=16B 满足 p5，故 p5 指向的地址为 0x403010。

(7) 画最终的堆结构图

堆结构图应该为：

P5：0X403010
P4：0X403020
P3：0X403030

参考答案：

p1 = 0x403010（内存片浪费 4 字节）
p2 = 0x403020（内存片浪费 4 字节）
p3= 0x403030

p4 = 0x403020（p2 释放的内存片刚好够用）
p5 = 0x403010（同上）

p5 执行 malloc，发现哑元和分配给 p4 的内存片之间的空闲区（16 字节）刚好够用，用该空闲区装新内存片（没有浪费空间），返回地址是 0x403010。

最终的堆结构图：