# exec 和 堆空间管理

同济大学计算机系　操作系统作业　　2023-12-7

学号　2154312　　姓名　郑博远

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);
}
```

答:　arg[0]:　showCmdParam

arg[1]:　arg1

arg[2]:　arg2

~~Over!~~ // exec 已经装了新的代码段，exit 结束后进程就要终止了（这个要特别注意！execv 不是普通的子程序调用）

此外还要注意，本题是父进程换了图像！

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) ; | (1) 指针 p1 的值是多少 ? |
| char *p2= malloc(4) ; | (2) 指针 p2 的值是多少 ? |
| char *p3= malloc(32) ; | (3) 指针 p3 的值是多少 ? |
| free(p2); | |
| char *p4= malloc(8) ; | (4) 指针 p4 的值是多少 ? |
| free(p1); | |
| char *p5= malloc(8) ; | (5) 指针 p5 的值是多少 ? (6) 情景分析 |
| | (7) 画最终的堆结构图 |

答:

(1) 0x403010（内存片浪费了 4 字节）

(2) 0x403020（内存片浪费了 4 字节）

(3) 0x403030

(4) 0x403020

(5) 0x403010

(6)

第一次执行 malloc 时，初始化堆空间，执行 brk 系统调用。malloc_head=0x403000，mallc_end = 0x406000。在 malloc_head 所指向的位置写一个 flist 哑元。之后，进行动态内存分配。此时，内存片链表只有一个哑元，不存在 2 个已分配内存片之间的空间区可以容纳新内存片的可能。所以，在内存片链表尾部（哑元的后面）追加一个长度是 16 字节的新内存片。malloc 函数的返回值 p1 是新内存片数据区的首部 0x403010。因为需要 8 字节对齐（所有 flist 的首地址必须是 8 的整数倍），所以新内存片的尾部浪费 4 字节。

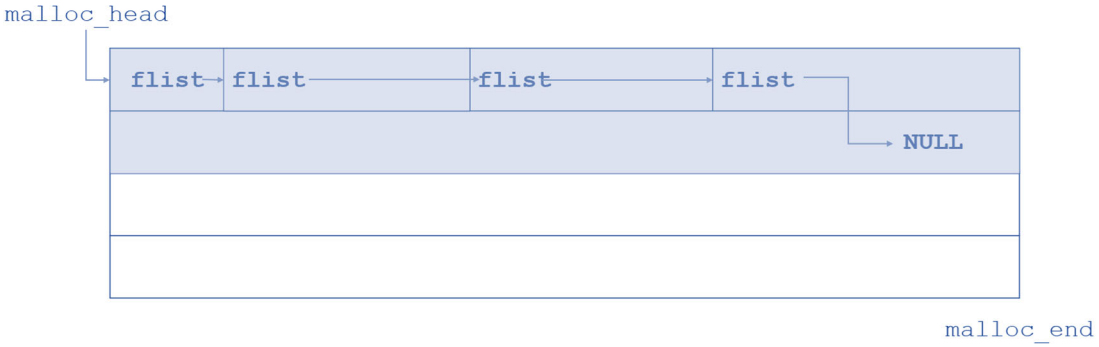
同理，p2 执行 malloc 时，在链表尾部追加一个长度是 16 字节的新内存片（含字节对齐），返回地址是 0x403020；p3 执行 malloc 时，在链表尾部追加一个长度是 32+8=40 字节的新内存片（不含字节对齐），返回地址是 0x403030。

此后，free 释放 p2 所指向的空间，将前一个 flist 的 nlist 字段指向其后一个 flist。之后，p4 执行 malloc，发现第一、二个内存片之间的空间区可以容纳新内存片的可能，因此在此之间紧贴着第一个内存片插入一个长度是 8+8=16 字节的新内存片（不含字节对齐），返回地址是 0x403020。

此后，free 释放 p1 所指向的空间，将前一个 flist 的 nlist 字段指向其后一个 flist。之后，p5 执行 malloc，发现第一、二个内存片之间的空间区可以容纳新内存片的可能，因此在此之间紧贴着第一个内存片插入一个长度是 8+8=16 字节的新内存片（不含字节对齐），返回地址是 0x403010。

答案正确:）

(7) 如下图所示：




（二、三、四，不必过于详细。要求掌握相关 PPT 的标题和主干步骤。）

三、简述 tryExec 进程创建子进程的过程

答：tryExec 通过钩子函数 fork，执行 2#号系统调用进行 fork。过程中，检查 Process 表含有空闲的 PCB，通过 NewProc()创建新的子进程。具体来说，为子进程分配空闲的 PCB，并复制相对虚实地址映射表，此后 NewProc 返回 0。此后，对于父进程来说，返回值为 0，将子进程的 pid 填到 EAX 中作为系统调用返回值；对于子进程来说，在 NewProc 栈帧上，通过 Swtch 函数返回，返回值为 1，将 0 填写到 EAX 中作为系统调用返回值。回到 fork 钩子函数后，再进一步返回到用户态程序中。

四、简述程序 showCmdParam 的加载过程

答：返回用户态后，新 fork 出的子进程进 else 分支调用 execv 钩子函数调用加载 showCmdParam。通过 execv 库函数清点 argc，并执行 11#号系统调用 exec。首先搜寻并解析可执行文件，然后将命令行参数复制到核心态空间的 fakeStack 中，接着释放原图像的正文、数据与堆栈段并重构 showCmdParam 的新正文段，然后重新构建正文、数据、堆栈段以及相对虚实地址映射表。最后将 fakeStack 复制到新构建的用户栈并清空用户态寄存器组。此后，返回用户态执行 runtime()驱动应用程序，从而进入 main1 函数执行。

五、简述子进程 showCmdParam 的终止过程

答：tryExec 与 showCmdParam 两个进程的终止先后顺序取决于调度顺序：

① 若 tryExec 先终止：调用 exit 系统调用，将 showCmdParam 转移为 1#进程的子程序。待 showCmdParam 终止时，调用 exit 系统调用，将 user 结构写入盘交换区，关闭所有文件、释放代码段、相对虚实映射表等，置终止状态。此后唤醒

父进程 1#进程，回收 showCmdParam 的 PCB。

② 若 showCmdParam 先终止：showCmdParam 终止时，调用 exit 系统调用，将 user 结构写入盘交换区，关闭所有文件、释放代码段、相对虚实映射表等，置终止状态。此时，其父进程未在睡眠也未终止，没有唤醒父进程。待父进程 tryExec 终止后，将 showCmdParam 的 ppid 设为 1#进程并唤醒 1#进程，此时 1#进程回收 showCmdParam 的 PCB。