得分

第四题(10分)

Bob 是一名刚刚学完异常的同学,他希望通过配合 kill 和 signal 的使用,能让两个进程向同一个文件中交替地打印出字符。可惜他的 tshlab 做得不过关,导致他写的这个程序有各种 BUG。你能帮帮他吗?

```
#include "csapp.h"
 2 | #define MAXN 6
 3
   int parentPID = 0;
   int childPID = 0;
 5
   int count = 1;
 6
   int fd1 = 1;
 7
   void handler1() {
 8
       if (count > MAXN)
 9
          return;
       for (int i = 0; i < count; i++)
10
11
          write(fd1, "+", 1);
12
                  Χ
13
       kill(parentPID, SIGUSR2);
14
15
   void handler2() {
16
       if (count > MAXN)
17
          return;
       for (int i = 0; i < count; i++)
18
19
          write(fd1, "-", 1);
                 Y
20
21
       kill(childPID, SIGUSR1);
22
23
24
   int main() {
25
       signal(SIGUSR1, handler1);
26
       signal(SIGUSR2, handler2);
27
       parentPID = getpid();
28
       childPID = fork();
29
       fd1 = open("file.txt", O RDWR);
```

注意:假设程序能在任意时刻被系统打断、调度,并且调度的时间切片大小是不确定的,可以足够地长。在每次程序执行前,file.txt是一个已经存在的**空文件**。

Part A. (1分) 此时, X 处语句和 Y 处语句都是 count++;, Z 处语句是空语句。Alice 测试该代码,发现有时 file.txt 中没有任何输出!请解释原因。(提示: 考虑 28 行语句 fork 以后,下一次被调度的进程,并从这个角度回答本题。不需要给出解决方案)

Part B. (6分) Bob 根据 Alice 的反馈,在某两行之间加了若干代码,修复了 Part A的问题。当 X 处代码和 Y 处代码都是 count++;、Z 处为空时,Bob 期望 file.txt 中的输出是:

+-++--+++----

可 Alice 测评 Bob 的程序的时候,却发现有时 Bob 的程序在 file.txt 中的输出是:

+-+--+---

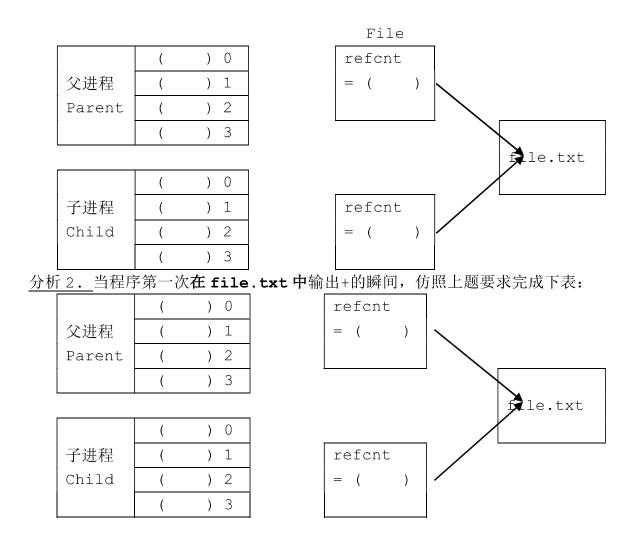
而与此同时,终端上出现了如下的输出:

+

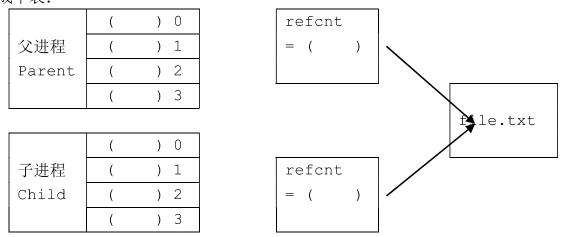
Bob 找不到自己的代码的 BUG,只好向 Alice 求助。Alice 帮他做了如下分析:分析 1. 当程序第一次**在终端**上输出+的瞬间,请完成下表。要求:

- (1) 在"描述符表"一栏中,用"√"勾选该进程当前 fd1 的值。
- (2) 在"打开文件表"一栏中,填写该项的 refcnt (即,被引用多少次)。如果某一项不存在,请在括号中写"0"(并忽略其指向 v-node 表的箭头)。
- (3) 画出"描述符表"到"打开文件表"的表项指向关系。不需要画关于标准输入/标准输出/标准错误的箭头。评分时不对箭头评分,**请务必保证前两步的解答与箭头的连接情况匹配。**

描述符表 打开文件表 v-node 表 Descriptor Open V-node



<u>分析 3.</u> 如果要产生 Bob 预期的输出,三级表的关系应当是什么? 仿照上题要求完成下表:



Part C. (2分) Bob 很高兴, 他知道 Part B 的代码是怎么错的了! 不过 Alice 仍然想考考 Bob。对于 Part B 的错误代码, 如果终端上输出的是+++, 那么

file.txt 中的内容是什么?请在下框中写出答案。

Part D. (1分) Bob 修复了 Part B的问题,使得代码能够产生预期的输出。现在,Bob 又希望自己的代码最终输出的是+--+++----,为此,他对 X、Y、Z 处做了如下的修改。X、Y 处语句已做如下填写,请帮助 Bob 补上 Z 处语句。

X 处填写为: count += 2;

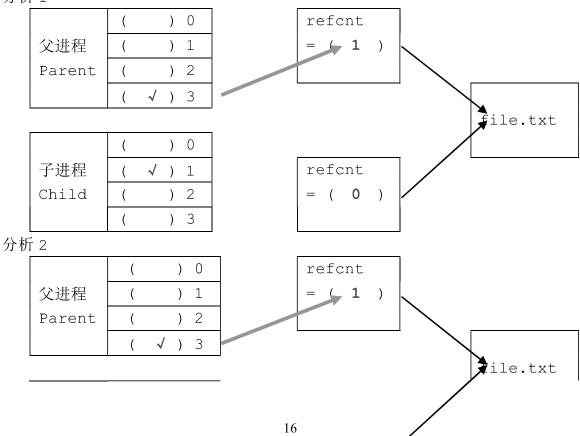
Y 处填写为: count += 2;

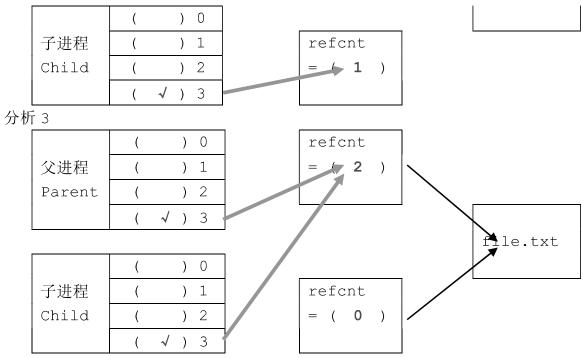
Z 处填写为:

【答案】Part A. 如果 28 行 fork 执行过后,子进程先被调度了,并且执行完 所有代码并退出,那么父进程的 kill 操作就无效了。

Part B.

分析 1





Part D. count = 2; (或 count++;等,只要让 count 最终的值是 2 就可以了)

【评分标准】

Part A. 意思对即可, 1分。

Part B. **打开文件表的两个条目可上下颠倒**。每个分析 2 分: 勾选对父进程的 fd1, 0.5 分; 勾对子进程的 fd1, 0.5 分; 两个 refcnt 各 0.5 分。由于箭头可以被 refcnt 确定,**因此对箭头的连接不赋分**。

Part C. 答案正确的 2 分。写对前 8 个字符的得 1 分,作为"辛苦分",因为能写对前 8 个字符表明理解这道题是怎么回事了,但是由于粗心而导致后面的模拟出错。

Part D. 1分。漏分号的不扣分。