

得分

第五题（10 分）

以下程序运行时系统调用全部正确执行，且每个信号都被处理到。请给出代码运行后所有可能的输出结果。

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

int c = 1;

void handler1(int sig) {
    c++;
    printf("%d", c);
}

int main() {

    signal(SIGUSR1, handler1);
    sigset_t s;
    sigemptyset(&s);
    sigaddset(&s, SIGUSR1);
    sigprocmask(SIG_BLOCK, &s, 0);

    int pid = fork()?fork():fork();

    if (pid == 0) {
        kill(getppid(), SIGUSR1);
        printf("S");
        sigprocmask(SIG_UNBLOCK, &s, 0);
        exit(0);
    } else {
        while (waitpid(-1, NULL, 0) != -1);
    }
}
```

```
        sigprocmask(SIG_UNBLOCK, &s, 0);  
        printf("P");  
    }  
    return 0;  
}
```

答:

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？

```

1  #include "csapp.h"
2  #define MAXN 6
3  int parentPID = 0;
4  int childPID = 0;
5  int count = 1;
6  int fd1 = 1;
7  void handler1() {
8      if (count > MAXN)
9          return;
10     for (int i = 0; i < count; i++)
11         write(fd1, "+", 1);
12     X
13     kill(parentPID, SIGUSR2);
14 }
15 void handler2() {
16     if (count > MAXN)
17         return;
18     for (int i = 0; i < count; i++)
19         write(fd1, "-", 1);
20     Y
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);

```

```
30     if (childPID) {
31         Z
32         kill(childPID, SIGUSR1);
33     }
34     exit(0);
35 }
```

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：

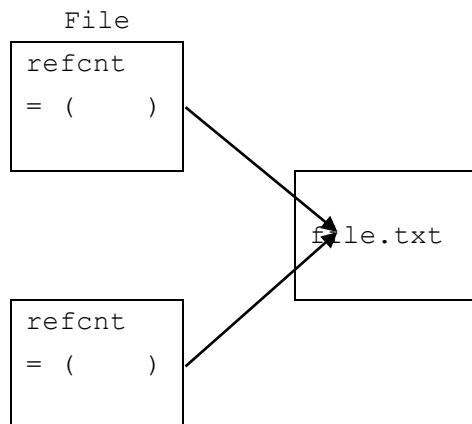
分析 1. 当程序第一次在终端上输出+的瞬间，请完成下表。要求：

- (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

父进程 Parent	() 0
	() 1
	() 2
	() 3

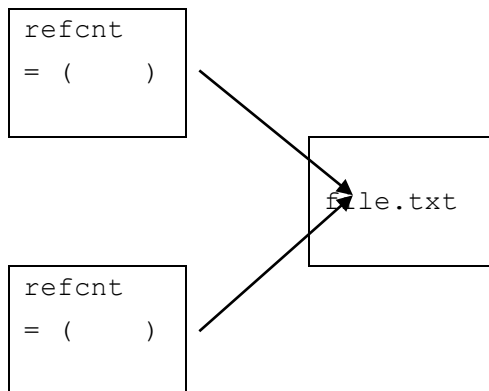
子进程 Child	() 0
	() 1
	() 2
	() 3



分析 2. 当程序第一次在 **file.txt** 中输出+的瞬间，仿照上题要求完成下表：

父进程 Parent	() 0
	() 1
	() 2
	() 3

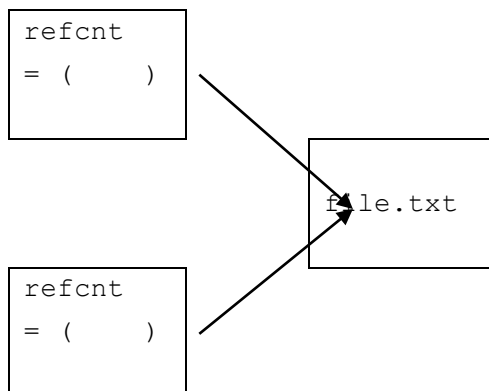
子进程 Child	() 0
	() 1
	() 2
	() 3



分析 3. 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：

父进程 Parent	() 0
	() 1
	() 2
	() 3

子进程 Child	() 0
	() 1
	() 2
	() 3



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

--

X 处填写为: `count += 2;`
Y 处填写为: `count += 2;`
Z 处填写为:

[illegible]