ICS 第十章

1. 假设缓冲区足够大，且 stdout 只有在关闭文件、换行与 fflush 的情况下才会刷新缓冲区。程序运行过程中的所有系统调用均成功。

(1)
```
int main() {
    printf("a");
    fork();
    printf("b");
    fork();
    printf("c");
    return 0;
}
```

(2)
```
int main() {
    write(1, "a", 1);
    fork();
    write(1, "b", 1);
    fork();
    write(1, "c", 1);
    return 0;
}
```

(3)
```
int main() {
    printf("a");
    fork();
    write(1, "b", 1);
    fork();
    write(1, "c", 1);
    return 0;
}
```

对于(1)号程序，写出它的一个可能的输出：_____。这个可能的输出是唯一的吗？_____。

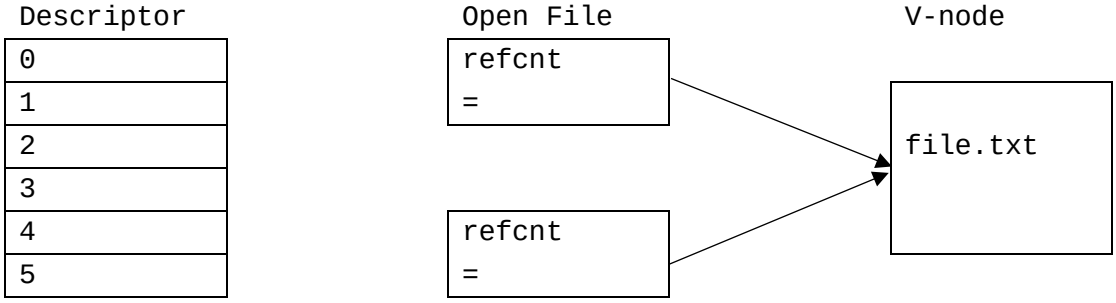
对于(2)号程序，它的输出中包含_____个 a，_____个 b，_____个 c。输出的第一个字符一定是_____。

对于(3)号程序，它的输出中包含_____个 a，_____个 b，_____个 c。输出的第一个字符一定是_____。

2. 假设磁盘上有空文件 file.txt。程序运行过程中的所有系统调用均成功。

```
int main() {
    int fd1 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
    int fd2 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
    printf("%d %d\n", fd1, fd2);
    write(fd1, "123", 3);
    write(fd2, "45", 2);
    close(fd1); close(fd2);
    return 0;
}
```

(1) 程序关闭 fd1 前，画出 LINUX 三级表结构。填写 Open File 表中的 refcnt。

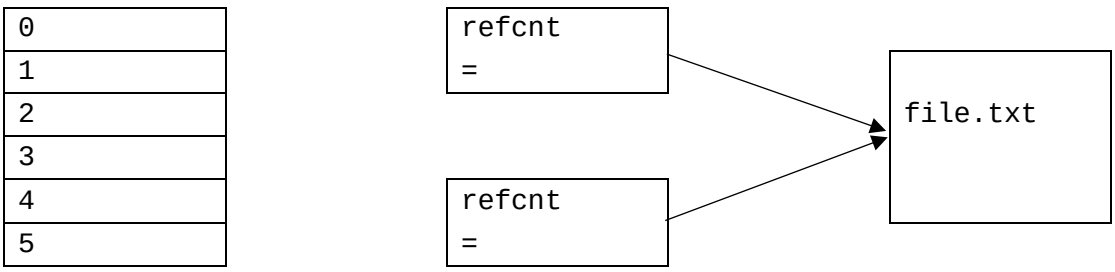


(2) 程序结束时，标准输出上的内容是_____，file.txt 中的内容是_____。

3. 假设磁盘上有空文件 file.txt。程序运行过程中的所有系统调用均成功。

```
int main() {
    int fd1 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
    int fd2 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
    dup2(fd2, fd1);
    printf("%d %d\n", fd1, fd2);
    write(fd1, "123", 3);
    write(fd2, "45", 2);
    close(fd1); close(fd2); return 0;
}
```

(1) 程序关闭 fd1 前，画出 LINUX 三级表结构。填写 Open File 表中的 refcnt。

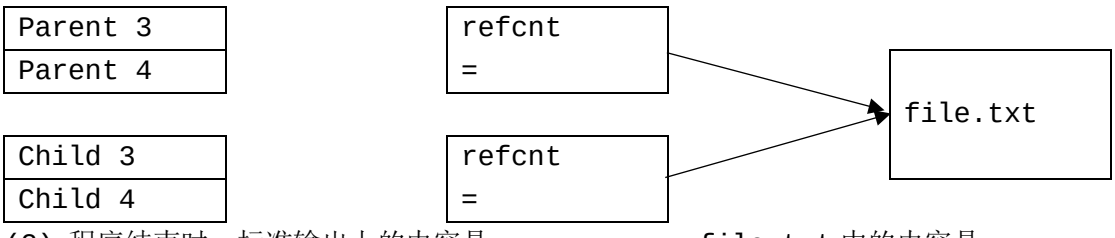


(2) 程序结束时，标准输出上的内容是_____，file.txt 中的内容是_____。

4. 假设磁盘上有空文件 file.txt。程序运行过程中的所有系统调用均成功。缓冲区足够大，且 stdout 只有在关闭文件、换行与 fflush 的情况下才会刷新缓冲区。

```
int main() {
    pid_t pid; int child_status;
    int fd1 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
    if ((pid = fork()) > 0) {            // Parent
        printf("P:%d ", fd1);
        write(fd1, "123", 3);
        waitpid(pid, &child_status, 0);
    } else {                             // Child
        printf("C:%d ", fd1);
        write(fd1, "45", 2);
    }
    close(fd1); return 0;
}
```

(1) 子进程关闭 fd1 前，画出 LINUX 三级表结构。填写 Open File 表中的 refcnt。



(2) 程序结束时，标准输出上的内容是_____，file.txt 中的内容是_____。

得分

（2015）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

（2018）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：

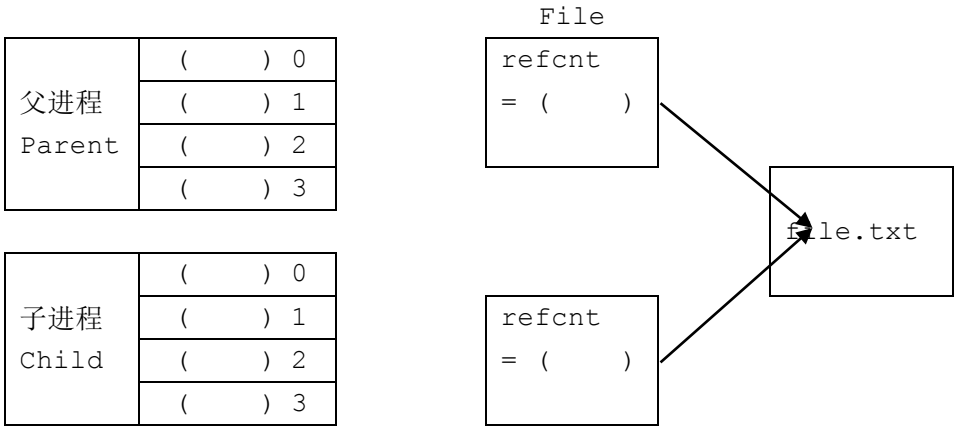
分析 1. 当程序第一次**在终端**上输出+的瞬间，请完成下表。要求：

(1) 在“描述符表”一栏中，用“√”勾选该进程当前 fd1 的值。

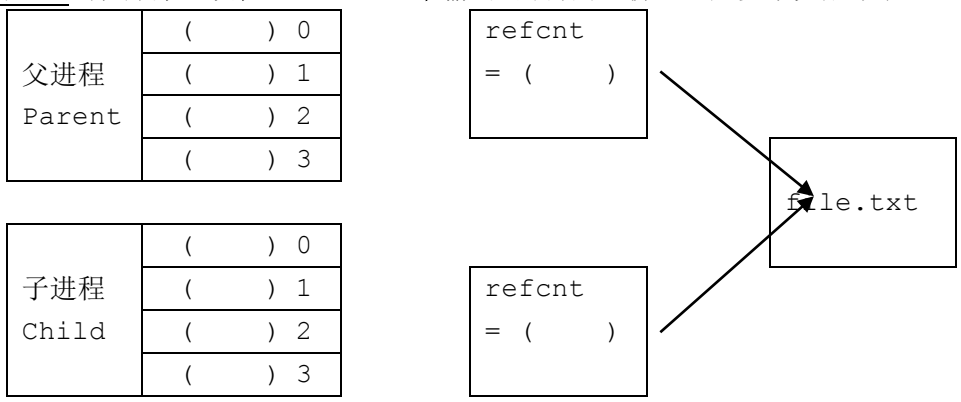
(2) 在“打开文件表”一栏中，填写该项的 refcnt（即，被引用多少次）。如果某一项不存在，请在括号中写“0”（并忽略其指向 v-node 表的箭头）。

(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。**

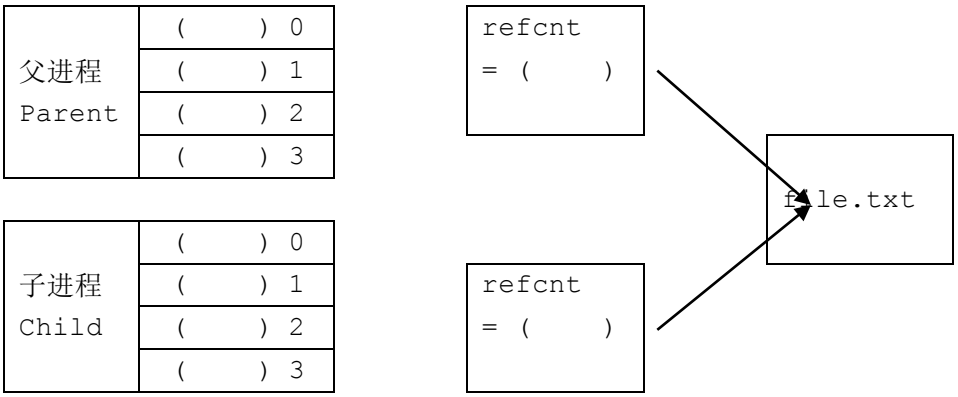
描述符表 Descriptor	打开文件表 Open	v-node 表 V-node



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：



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：



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：
