# 计算机系统概论（2024 秋）作业 5

## 1 fork()

阅读代码，回答以下问题

```
#include <cstdlib>
#include <unistd.h>
#include <cstdio>
int counter = 0;
int main(){
  for (int i = 0;i < 2; i++) {
    fork();
    counter++;
    printf("counter = %d\n", counter);
  }
  // 注意：这里没有 counter++;
  printf("counter = %d\n", counter);
  return 0;
}
```

### 1.1 程序会输出多少行？(空行不计算在内)

*10* 行, 5pt

### 1.2 程序的全部输出中，第一行和最后一行分别会是什么？

$counter = 1$; $counter = 2$, 5pt

### 1.3 根据系统对进程的调度情况，程序一共有多少种可能的输出结果？(注：如果同一时间有若干进程在同时运行，他们运行的先后顺序的不同可能导致输出结果不同)

*5 种，第二个 counter = 1 可以在 01234 个 counter = 2 之后*, 5pt

## 2 文件读写

- 假设文件 `file1.txt` 中有一个字符串 `aabbccdd` 。
- 下列 C 文件分别被编译成 `./program1` 和 `./program2`

```c
/* Program 1 */
#include <fcntl.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char const *argv[]) {
    int pid, fd_x, fd_y, fd_z;
    char buf[8];

    fd_x = open("file1.txt", O_RDWR);
    fd_y = open("file1.txt", O_RDWR);
    fd_z = open("file1.txt", O_RDWR);

    read(fd_x, buf, 2);
    read(fd_y, buf + 2, 4);

    if ((pid = fork()) == 0) {
        dup2(fd_x, STDOUT_FILENO);
        dup2(fd_y, STDIN_FILENO);
        execl("program2", "program2", NULL);
    }

    wait(NULL);

    read(fd_y, buf + 6, 2);
```

```
    write(fd_z, buf + 6, 2);
    write(fd_x, buf + 4, 2);
    write(fd_x, buf + 2, 2);

    close(fd_x);
    close(fd_y);
    close(fd_z);

    return 0;
}
```

```
/* Program 2 */
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char const *argv[]) {
    char buf2[2];
    read(STDIN_FILENO, buf2, 2);
    write(STDOUT_FILENO, buf2, 2);
}
```

对所调用的一些方法的解释
- open(filename, O_RDWR)：打开一个已有文件 filename，对其进行读写操作，起始位置为 0；
- execl(exename,...)：执行 exename 程序，与我们讲的 execve() 功能类似。

请注意不同时刻各个 fd 的访问位置

### 2.1 当执行 ./program1 后，文件 file1.txt 的内容是什么？

*ddccbbaa*, 5pt

### 2.2 从以下几点解释一下为什么是这个结果（言之有理即可，简短作答）：

共 15pt

- **./program2 做了什么？**

  从 *STDIN* 读两个字节，写到 *STDOUT*, 2pt

- **./program1 在 fork() 之前做了什么，相应的 buf 的内容如何变化？**

  从 *fd_x* 读 *2* 字节 *aa* ，保存到 *buf* ，*buf = aa*

  从 *fd_y* 读 *4* 字节 *aabb* ，保存到 *buf+2* ，*buf = aaaabb*, 4pt

- **./program1 调用 fork() 了以后，子进程在干什么？**

  把 *STDIN* 重定向到 *fd_y* ,*STDOUT* 重定向到 *fd_x* ,执行 *./program2*

  *./program2* 从 *fd_y* 读 *2* 字节 *cc* ,写入 *fd_x* ,*file1.txt* 变为 *aaccccdd*, 4pt

- **子进程返回后，./program1 又做了什么，相应的 buf 的内容如何变化？**

  从 *fd_y* 读 *2* 字节 *dd* ，保存到 *buf+6* ，*buf = aaaabbdd*

  把 *buf+6* 处 *2* 字节 *dd* 写入 *fd_z* ，*file1.txt* 变为 *ddccccdd*

  把 *buf+4* 处 *2* 字节 *bb* 写入 *fd_x* ，*file1.txt* 变为 *ddccbbdd*

  把 *buf+2* 处 *2* 字节 *aa* 写入 *fd_x* ，*file1.txt* 变为 *ddccbbaa*, 5pt

# 3 信号量

- 老师的办公室有一个空的白板。
- 老师会在白板为空时往白板上写一道物理题或一道化学题。
- 如果是一道物理题，喜爱物理的小 A 会解答出这道题并把题目擦掉
- 如果是一道化学题，喜爱化学的小 B 会解答出这道题并把题目擦掉

**请使用信号量和 P、V 原语，实现老师、小 A、小 B 三者的同步。**

```
sem_t board;      // 白板是否为空
sem_t physics;    // 白板上是否为物理题
sem_t chemistry;  // 白板上是否为化学题

void init() {
  Sem_init(&board, 0, ___(A) ___);
  Sem_init(&physics, 0, ___(B) ___);
  Sem_init(&chemistry, 0, ___(C) ___);
}
void teacher() {
  while (1) {
    Course c = (rand() & 1) ? PHYSICS : CHEMISTRY;
    ____(D) ____;
    在白板上写题目;
    if (c == PHYSICS) {
      ____(E) ____;
    } else {  // c == CHEMISTRY
      ____(F) ____;
    }
  }
}
void studentA() {
  while (1) {
    P(___(G) ___);
    解答物理题, 将其擦掉;
    V(___(H) ___);
  }
}
void studentB() {
  while (1) {
    P(___(I) ___);
    解答化学题, 将其擦掉;
    V(___(J) ___);
  }
}
```

(A) 1

(B) 0
(C) 0
(D) P(&board)
(E) V(&physics)
(F) V(&chemistry)
(G) &physics
(H) &board
(I) &chemistry
(J) &board, 每空 3pt, 共 30 pt

# 4 线程与子线程

- 阅读程序写结果，要求列出所有可能输出。
- 不考虑进程/线程创建失败的情况
- 可假设 `printf` 的输出不会被其他 `printf` 打断。

## 4.1

```c
#include <pthread.h>
#include <stdio.h>
int a = 0;
void* test(void* ptr) { a++; return NULL; }
int main() {
    pthread_t pid;
    pthread_create(&pid, NULL, test, NULL);
    pthread_join(pid, NULL);
    printf("a=%d\n", a);
    return 0;
}
```

a=1, 3pt

### 4.2

```c
#include <unistd.h>
#include <stdio.h>
int a = 0;
void test() { a++; }
int main() {
    int pid = fork();
    test();
    printf("a=%d\n", a);
    return 0;
}
```

a=1 \n a=1, 3pt

### 4.3

```c
#include <pthread.h>
#include <stdio.h>
int a = 0;
void* test(void* ptr) { a++; return NULL; }
int main() {
    pthread_t pid1, pid2;
    pthread_create(&pid1,NULL,test, NULL);
    pthread_create(&pid2,NULL,test, NULL);
    pthread_join(pid1, NULL);
    pthread_join(pid2, NULL);
    printf("a=%d\n", a);
    return 0;
}
```

a=1 or a=2, 4pt

### 4.4

```c
#include <unistd.h>
#include <stdio.h>
int a = 0;
void test() { a++; }
int main() {
    int pid = fork();
    if (pid != 0) pid = fork();
    test();
    if (pid != 0) test();
    printf("a=%d\n", a);
    return 0;
}
```

a=1 \n a=1 \n a=2 或 a=1 \n a=2 \n a=1 或 a=2 \n a=1 \n a=1, 15pt

## 5 信号处理

考虑如下程序：

```c
void handler (int sig) {
    printf("D");
    exit(4);
}
int main() {
    int pid, status;
    signal(SIGINT, handler);
    printf("A");
    pid = fork();
    printf("B");
    if (pid == 0) {
        printf("C");
    } else {
        kill(pid, SIGINT);
        waitpid(pid, &status, 0);
        printf("%d", WEXITSTATUS(status));
    }
    printf("E");
    exit(7);
}
```

**以下哪些是可能的输出结果（多选）：**

**A.** `ABCBE7E`

**B.** `ABD7E`

**C.** `ABBCE4E`

**D.** `ABCDB4E`

**E.** `ABBD4E`

*A*,*E*, 10pt