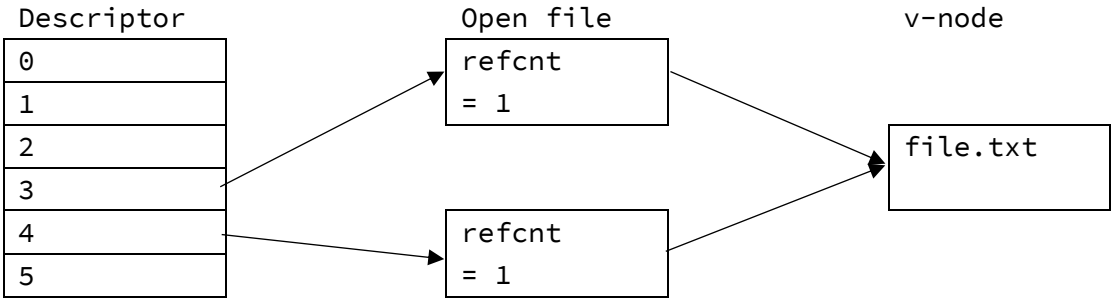
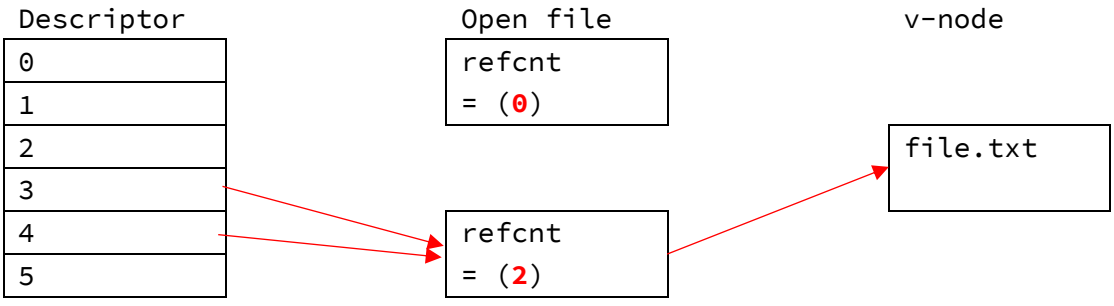
1. 假设磁盘上有空文件 file.txt。程序运行过程中的所有系统调用均成功。

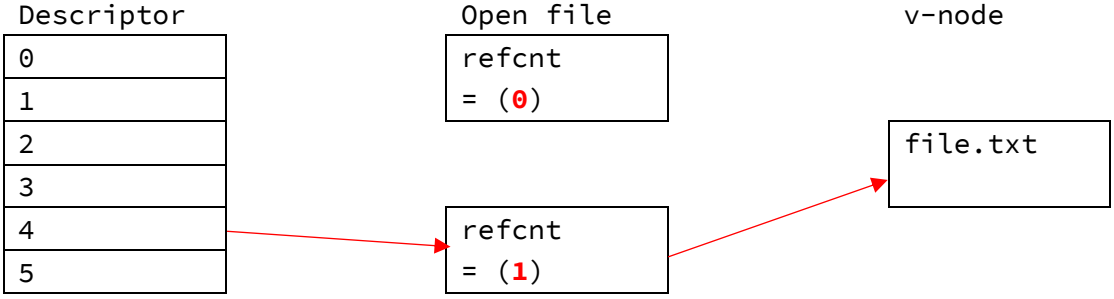
```
int main() {
   int fd1 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
   int fd2 = open("file.txt", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR);
   printf("%d %d\n", fd1, fd2);
   // A
   write(fd1, "012", 3);
   write(fd2, "ab", 2);
   dup2(fd2, fd1);
   // B
   write(fd1, "123", 3);
   write(fd2, "cd", 2);
   close(fd1);
   // C
   close(fd2);
   return 0;
}
```

已知在程序执行到 A 处时，画出 Linux 三级表的结构如下：



(1)请画出程序在执行到 B,C 处时 Linux 三级表的结构





(2)程序结束时，标准输出上的内容是_**3 4**_，file.txt 中的内容是_**ab123cd**_.

2. 判断以下说法的正确性

(√)	目录(directory)是一种特殊的文件，包含一组链接(link)，每个链接将一个文件名映射到一个文件。
(×)	关闭一个已经关闭的描述符时，不会出错
(√)	在进程调用 fork()之后，子进程会继承父进程的描述符表(file descriptor table)，也会继承 stdio 的缓冲区
(√)	在编写网络程序时，应该使用 Unix I/O 而不是标准 I/O

3. 假设某进程有且仅有五个已打开的文件描述符：0~4，分别引用了五个不同的文件，尝试运行以下代码：

```
dup2(3,2); dup2(0,3); dup2(1,10); dup2(10,4); dup2(4,0);
```

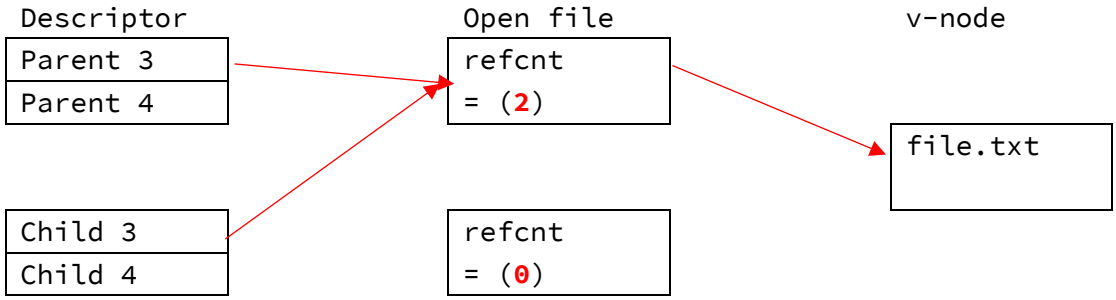
关于得到的结果，说法正确的是：**A**

A. 运行正常完成，现在有四个描述符引用同一个文件
B. 运行正常完成，现在进程共引用四个不同的文件
C. 由于试图从一个未打开的描述符进行复制，发生错误
D. 由于试图向一个未打开的描述符进行复制，发生错误

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)在子进程关闭 fd1 前，画出 Linux 三级表的结构如下



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

C:3 P:3 ;12345 或 45123 注意:C 一定在 P 前输出

5. 根据本课程介绍的 Intel x86-64 存储系统，填写表格中某一个进程从用户态切换至内核态时，和进程切换时对 TLB 和 cache 是否必须刷新 **A，Cache 一般情况下**

都不用刷新；TLB 需要在地址空间进行变化时刷新。

A．①不必刷新 ②不必刷新 ③刷新 ④不必刷新
B．①不必刷新 ②不必刷新 ③不必刷新 ④不必刷新
C．①刷新 ②不必刷新 ③刷新 ④刷新
D．①刷新 ②不必刷新 ③不必刷新 ④刷新

6．下列关于虚存和缓存的说法中，正确的是： D
A．TLB 是基于物理地址索引的高速缓存
B．多数系统中，SRAM 高速缓存基于虚拟地址索引
C．在进行**线程**切换后，TLB 条目绝大部分会失效
D．多数系统中，在进行进程切换后，SRAM 高速缓存中的内容不会失效

7．对于虚拟存储系统,一次访存过程中,下列命中组合不可能发生的是 D，TLB 命中时，页一定在物理内存中.
A.TLB 未命中，Cache 未命中，Page 未命中
B.TLB 未命中,Cache 命中， Page 命中
C.TLB 命中， Cache 未命中，Page 命中
D.TLB 命中， Cache 命中， Page 未命中

8．关于写时复制（copy-on-write，COW）技术的说法，不正确的是：
A．写时复制既可以发生在父子进程之间，也可以发生在对等线程之间
B．写时复制既需要硬件的异常机制，也需要操作系统软件的配合
C．写时复制既可以用于普通文件，也可以用于匿名文件
D．写时复制既可以用于共享区域，也可以用于私有区域
A/D。未解之谜。如果将共享区域理解为 MAP_SHARED 则 D 错误；否则共享库也是一种共享区域，会发生 COW。如果参照 Wikipedia 上的解说，COW 是一种技术，用户可以自行实现，那么对等线程之间也可能发生 COW（例如 string y=x 之后修改 y，C++ 中用 COW 实现）。 A 可能是更好的选择。

9．假设有一台 64 位的计算机的物理页块大小是 8KB,采用三级页表进行虚拟地址寻址,它的虚拟地址的 VPO(Virtual Page Offset,虚拟页偏移)有 13 位,问它的虚拟地址的 VPN(Virtual Page Number,虚拟页号码)有多少位?C
A．20
B．27
C．30
D．33
本题考查对页表组成的理解.页块大小为 8KB,即 2^13 byte.在 64 位机器上,一个页表条目为 8byte.故共有页表条目 2^10 项,故每一级页表可以表示 10 位地址.因此三级页表存储，共需要 10*3=30 位.

10．进程 P1 通过 fork()函数产生一个子进程 P2.假设执行 fork()函数之前，进程 P1 占用了 53 个(用户态的)物理页,则 fork 函数之后,进程 P1 和进程 P2 共占用_____个(用户态的)物理 页;假设执行 fork()函数之前进程 P1 中有一个可读写的物理页,则执行 fork()函数之后,进程 P1 对该物理页的页表项权限为 B，fork 的时候因为 copy-on-write 机制，并不会占用额外的物理页，同时把权限标记成只读，这样

write 时会触发 page fault(PF),操作系统负责复制.

A. 53,读写

B. 53,只读

C. 106,读写

D. 106,只读

11. Intel 的 IA32 体系结构采用二级页表,称第一级页表为页目录(Page Directory),第二级页表为页表(Page Table)。页面的大小为 4KB,页表项 4 字节。以下给出了页目录与若干页表中的部分内容,例如,页目录中的第 1 个项索引到的是页表 3,页表 1 中的第 3 个项索引到的是物理地址中的第 5 个页。则十六进制逻辑地址 8052CB 经过地址转换后形成的物理地址应为十进制的(**B**)。

页目录		页表 1		页表 2		页表 3	
VPN	页表号	VPN	页号	VPN	页号	VPN	页号
1	3	3	5	2	1	2	9
2	1	4	2	4	4	3	8
3	2	5	7	8	6	5	3

A. 21195

B. 29387

C. 21126

D. 47195

4KB=2^12,所以页内地址有 12 位.4KB/4B=1K,所以页目录和每个页表中的页表项数为 1K 个.因此,在 32 位的虚拟地址中,最低的 12 位为页内地址(Offset),最高的 10 位为页目录的虚拟地址(Dir),中间 10 位为页表的虚拟地址(Table).

十六进制逻辑地址 8052CB 转换为二进制后为 100000000101 001011001011,Dir 为 10,即 10 进制的 2,在表中对应到页表 1.

Table 为 101,即 10 进制的 5,在表中对应到物理页面 7.因此,物理地址应为 7 的二进制 111 和 Offset 的拼合,即 111001011001011,转换为十进制为 29387,答案为 B.

12. 假定整型变量 A 的虚拟地址空间为 0x12345cf0,另一整形变量 B 的虚拟地址 0x12345d98,假定一个 page 的长度为 0x1000 byte,A 的物理地址数值和 B 的物理地址数值关系应该为: **B,在同一页中**

A.A 的物理地址数值始终大于 B 的物理地址数值

B.A 的物理地址数值始终小于 B 的物理地址数值

C.A 的物理地址数值和 B 的物理地址数值大小取决于动态内存分配策略

D.无法判定两个物理地址值的大小

13. 下列与虚拟内存有关的说法中哪些是不对的?

A.操作系统为每个进程提供一个独立的页表,用于将其虚拟地址空间映射到物理地址空间.

B.MMU 使用页表进行地址翻译时,虚拟地址的虚拟页面偏移与物理地址的物理页面偏移是相同的.

C.若某个进程的工作集大小超出了物理内存的大小,则可能出现抖动现象.

D.动态内存分配管理,采用双向链表组织空闲块,使得首次适配的分配与释放均是空闲块数量的线性时间.

D，malloclab 的实现一般释放是 O(1)的

14. 在 Core i7 中,关于虚拟地址和物理地址的说法,不正确的是:

A.VPO = CI + CO

B.PPN = TLBT + TLBI

C.VPN1 = VPN2 = VPN3 = VPN4

D.TLBT + TLBI = VPN

B，参照下图进行理解,物理地址通常没有 48 位这么长；让 CI+CO=VPO 是为了在进行地址翻译和取数据时更好的并行性

63-48 位	VPN				VPO		虚拟地址
63-48 位	VPN1	VPN2	VPN3	VPN4	VPO		虚拟地址
63-48 位	TLBT			TLBI	CI	CO	虚拟地址
63-52 位	CT=PPN				CI	CO	物理地址

15. 已知某系统页面长 8KB,页表项 4 字节,采用多层分页策略映射 64 位虚拟地址空间.若限定最高层页表占 1 页,则它可以采用多少层的分页策略?

A.3 层 B.4 层 C.5 层 D.6 层

C.由题意,64 位虚拟地址的虚拟空间大小为 264.页面长为 8KB,页表项 4 字节,所以一个页面可存放 2K 个表项.由于最高层页表占 1 页,也就是说其页表项个数最多为 2K 个,每一项对应一页,每页又可存放 2K 个页表项,依次类推可知,采用的分页层数为:5 层.

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：**共 3 种： S2PS2P SS2P2P S2SP2P**

得分

第六题（15 分）

为了提升虚拟内存地址的转换效率，降低遍历两级页表结构所带来的地址转换开销，英特尔处理器中引入了大页 TLB，即一个 TLB 项可以涵盖整个 4MB 对齐的地址空间（针对 32 位模式）。只要设置页目录页中页目录项（PDE）的大页标志位，即可让 MMU 识别这是一个大页 PDE，并加载到大页 TLB 项中。大页 PDE 中记录的物理内存页面号必须是 4MB 对齐的，并且整个连续的 4MB 内存均可统一通过该大页 PDE 进行地址转换。

在 32 位的 Linux 系统中，为了方便访问物理内存，内核将地址 0~768MB 间的物理内存映射到虚拟内存地址 3GB~3GB+768MB 上，并通过大页 PDE 进行进行该区间的地址转换。任何 0~768MB 的物理内存地址可以直接通过加 3G（0xC0000000）的方式得到其虚拟内存地址。在内核中，除了该区间的内存外，其他地址的内存通常都通过普通的两级页表结构来进行地址转换。

假设在我们使用的处理器中有 2 个大页 TLB 项，其当前状态如下：

索引号	TLB 标记	页面号	有效位
0	0xC48	0x04800	1
1	0xC9C	0x09C00	1

有 4 个普通 TLB 项，当前的状态如下：

索引号	TLB 标记	页面号	有效位
0	0xF8034	0x04812	1
1	0xF8033	0x09812	1
2	0xF4427	0x12137	1
3	0xF44AE	0x17343	1

当前页活跃的目录页（PD）中的部分 PDE 的内容如下：

PDE 索引	页面号	其他标志	大页位	存在位
786	0x04800	…	1	1

807	0x09C00	…	1	1
977	0x09C33	…	0	1
992	0x09078	…	0	1

注：普通页面大小为 4KB，并且 4KB 对齐。每个页面的页面号为其页面起始物理地址除以 4096 得到。大页由连续 1024 个 4KB 小页组成，且 4MB 对齐。

注：严格来讲，本题的题目在 TLB tag 的一列有误。以第一问中访问地址 0xC48012024 为例，大页 TLB 有两组，组索引占 1 位，此时 TLB tag 按理来说不应该为 0xC48，而应该为（0xC48 >> 1）=0x624。其他的 TLB tag 同理。不过本题介绍了大页概念，而且难度适宜，TLB tag 的错误也不太影响解题，总体来说还是一道好题，难以割舍因而选入。

1．分析下面的指令序列，

```
movl $0xC48012024, %ebx
movl $128, (%ebx)
movl $0xF8034000, %ecx
movl $36(%ecx), %eax
```

请问，执行完上述指令后，eax 寄存器中的内容是（ **128** ）；在执行上述指令过程中，共发生了（ **0** ）次 TLB miss？同时会发生（ **0** ）次 page fault？

注：不能确定时填写“--”。

TLB 命中时，不会出现 page fault；两个虚拟地址的物理地址相同

2．请判断下列页面号对应的页面中，哪些一定是页表页？哪些不是？哪些不确定？

页面号	是否为页表页（是/不是/不确定）
0x04800	**不确定，任何页面都可能是页表页**
0x09C33	**是，是当前页目录对应的页表项**
0x09812	**不确定，任何页面都可能是页表页**

3．下列虚拟地址中哪一个对应着够将虚拟内存地址 0xF4427048 映射到物理内存地址 0x14321048 的页表项（ **B** ）？

(A) 0x09C33027　　(B) 0xC9C3309C

(C) 0xC9C33027　　(D) 0x09C3309C

通过上述虚拟地址，利用 movl 指令修改对应的页表项，完成上述映射，在此过程中，是否会产生 TLB miss？（ **不会** ）（回答：会/不会/不确定）

修改页表项后，是否可以立即直接使用下面的指令序列将物理内存地址 0x14321048 开始的一个 32 位整数清零？为什么？

```
movl $0xF4427048, %ebx
movl $0, (%ebx)
```

答：**不能，因为此时 TLB 项和页表项不一致。应该先把 TLB 项置为失效，然后通过 TLB miss 重新缓存页表**