考试科目名称 操作系统（A 卷 参考答案）

2022——2023 学年第 2 学期 教师_____ 考试方式：闭卷

系（专业）_____ 年级_____ 班级_____

学号_____ 姓名_____ 成绩_____

题号	一	二
分数		

得分	

一、综合题（共 42 分）

1. UNIX 系统中，运行下列代码，假设该代码运行过程中所创建的进程 id 按创建先后顺序依次分配为 1,2,3,...，shell 进程 id 为 0：

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>

int count = 0;

int main(void)
{
    setbuf(stdout, NULL); // 禁用 printf 缓存

    // 若为子进程，则 fork 返回值为 0；若为父进程，fork 返回值为新创建的子进程 id
    count += fork();
    count += fork();
    count += fork();

    // getppid() 返回当前进程父进程id，getpid() 返回当前进程id
    printf("%d-%d: %d\n", getppid(), getpid(), count);

    wait(NULL); // 等待子进程结束

    return 0;
}
```

试回答如下问题（10 分）：

（一）此代码运行过程中共产生多少个进程，画出所有进程已创建且未结束时的进程树（以 Shell 进程为根节点）？

（二）假设所有系统调用均执行成功，请给出代码的一个可能执行结果（**留意回车换行等细节**）；

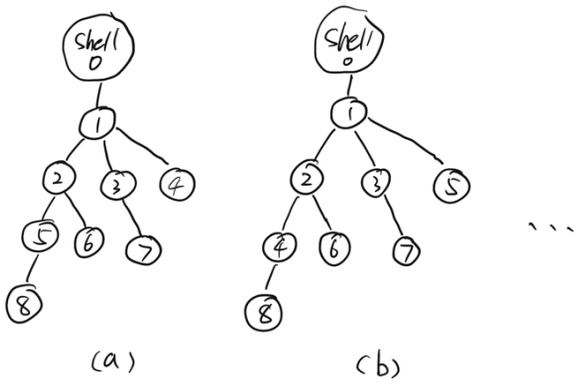
（三）在所有系统调用均执行成功的情况下，上述代码运行是否可能有多种结果，为什么？

（四）代码中哪些 API 函数会触发 I/O 相关的系统调用，哪些 API 函数会触发进程控制和进程同步的系统调用？

（五）如果不使用上述 API 函数，是否能直接进行相关系统调用？如果可以，试讨论这样做的弊端？

参考答案：

（一）共创建 8 个进程（main 编号为 1，长子进程为 2，其他子孙进程编号依次为 3-8，但编号不定），可有多种结果，给出两种常见示例（大部分同学的答案落在此两种）：



（二）输出示例跟第（一）题相关（每行值须一致，但行与行顺序自由）；

(a):
0-1: 9
1-2: 11
1-3: 9
1-4: 5
2-5: 8
2-6: 5
3-7: 2
5-8: 0

(b):
0-1: 10
1-2: 10
1-3: 9
2-4: 8
1-5: 5
2-6: 4
3-7: 2
4-8: 0

（三）是，父子进程调度执行的**先后顺序不定**，例如长子进程可能在次子进程之后执行 fork 系统调用；不同进程执行 printf 的先后顺序不定；

（四） I/O 相关：setbuf, printf; 进程控制相关：fork, wait, getppid, getpid**（各列出一个即可）**

（五）可以，但代码复杂度增加，**可读性差**；需要关注不同硬件平台、不同操作系统之间的差异，**可移植性差**。

2. 一个多道批处理系统，用户可使用的主存为 200KB，采用可变分区存储管理技术，分区回收时进行相邻空闲分区合并，作业调度采用先来先服务算法，进程调度采用时间片轮转法（时间片远小于各作业运行时间），各作业具体情况如下图所示：

作业名	到达时间	估计运行时间（分钟）	主存需求(KB)
Job1	9:00	15	70
Job2	9:10	35	100
Job3	9:20	25	20
Job4	9:30	30	50
Job5	9:45	20	30

试回答如下问题（6 分）：

（一） 若空闲区分配算法为最优适应，列出各作业**创建进程时间**与结束时间，以及主存用户区的变化情况（**按关键时间节点列出**）；

（二） 若空闲区分配算法为最坏适应，列出各作业**创建进程时间**与结束时间，以及主存用户区的变化情况（**按关键时间节点列出**）；

（三） 可变分区存储管理技术是否存在“内碎片”和“外碎片”现象？请列举一个既存在“内碎片”又存在“外碎片”的存储管理技术，列举一个仅存在“内碎片”的存储管理技术。

参考答案：

（一）

	创建进程时间	结束时间
Job1	9:00	9:20
Job2	9:10	10:45
Job3	9:20	10:30
Job4	9:30	10:55
Job5	10:30	11:05

内存图略！

（二）

	创建进程时间	结束时间
Job1	9:00	9:20
Job2	9:10	11:00
Job3	9:20	10:45
Job4	9:30	11:05
Job5	9:45	11:00

内存图略！

（三）可变分区**仅存在“外碎片”**；**伙伴系统分配**，既存在“外碎片”又存在“内碎片；**固定分区、分页**，仅存在“内碎片”

3. 内核代码的临界区管理往往会用到自旋锁，为何在单核单处理器系统中自旋锁的加锁和释放锁操作可以简单实现为关中断和开中断？多核多处理器系统中上述方法为何失效？如果存在一条 XCHG 指令可以在内存单元和寄存器之间交换值，请给出基于此指令的自旋锁实现思路(包括 **lock, unlock**)。（3 分）

参考答案：

单核单处理器，关中断即意味**代码并发不会发生（代码串行执行）**，可避免多个执行代码进入相关临界区

多核多处理器，关中断只能使当前核（处理器）上执行的代码串行化，但不能阻止其他核（处理器）上执行相关临界区代码，**无法阻止代码并行执行**

```
enter_region:
        TSL REGISTER,LOCK          | copy lock to register and set lock to 1
        CMP REGISTER,#0            | was lock zero?
        JNE enter_region           | if it was not zero, lock was set, so loop
        RET                        | return to caller; critical region entered

leave_region:
        MOVE LOCK,#0               | store a 0 in lock
        RET                        | return to caller
```

类似上述代码（伪代码）均可，或者注释表达意思合适!!!

4. 若某个系统中有 5 个并发进程分别是 P0、P1、P2、P3、P4，四类资源分别标记为 A、B、C、D，系统目前各进程的资源分配、申请情况如下表所示：

	Allocation				Request				Available			
Process	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	0	1	1	0	0	0	1	1	0	0
P1	2	0	0	0	1	0	0	0				
P2	0	2	2	2	2	0	1	0				
P3	2	1	1	1	1	0	0	2				
P4	0	3	2	2	0	2	0	0				

试回答如下问题（3 分）：

（一） 该系统目前是否存在死锁？若存在死锁，涉及哪些进程？

（二） 如果要避免死锁发生，操作系统需要在什么时刻进行决策，还需要掌握什么信息？

参考答案：

（一） 存在**死锁**，涉及 **P2、P3、P4 进程**

（二） 在资源申请和分配时，需要掌握各进程目前已分配的各种资源，及对各种资源需求的最大量

5. 一个 32 位系统的计算机，具有 2GB 物理内存，其上的操作系统采用请求分页存储管理技术，页面大小为 1KB，页表项大小为 4B。系统为某进程固定分配了四个页框，当前时刻为逻辑时钟 164，页面访问情况如下表所示，试回答如下问题（7 分）：

页号	页框号	装入时间	最近访问时间	访问位	修改位
0	25	60	161	0	1
1	102	120	160	0	0
2	55	42	162	1	1
3	7	56	163	1	0

（一） 该系统中一个进程页表的理论最大尺寸是多少字节？如果采用二级页表机制，那么 32 位的逻辑地址该如何划分（**页目录位移、页表页位移、页内位移分别占多少位**）？试简要讨论多级页表机制的优缺点。

（二） 如果采用反置页表，则该系统的**反置页表**表项数是多少？每个反置页表表项需要包含的必要信息是什么？试简要讨论其优缺点。

（三） 试解释为何 0 号页面对应的页表项，其访问位是 0，修改位是 1？

（四） 此时**栈指针**指向 4200（十进制）处，若要执行指令 **CALL 2100**（十进制），会涉及几次内存访问？分别访问哪些页面？试分析采用以下几种不同的页面替换算法：先进先出、时钟页面、最近不使用，发生缺页中断的次数分别是多少？

（五） 针对（四）中出现的一条指令执行可能引发多次缺页中断的情况，试着给出一种软硬件配合的机制用于尽量减少缺页中断发生的情况（**提示：锁定位**）

参考答案：

（一）**2^24B (16MB)**；**(14bits, 8bits, 10bits)**；优点：页表不需要连续存储在内存中，使得未使用的页面对应的页表项可不占内存空间，缺点：增加地址转换的时间开销（多次访问内存）

（二）2GB/1KB = **2^21** 项；（进程号，页号）；优点：节约内存，缺点：地址转换效率低

（三）该页面之前被修改过，尚未写回磁盘，故修改位为 1，引用位在修改发生后经历至少一次操作内核的清零操作

（四）指该指令**执行过程**中（**指令已取**），涉及 **2 次**内存访问，分别访问 (4200/1024) **4 号**和 (2100/1024) **2 号**页面；

先进先出：2 次；时钟页面：1 次；最近不使用：1 次

（五）硬件解释 CALL 指令过程中，先期将已在内存中指令涉及的页面对应的页表项锁定位置 1，当操作系统内核页面替换算法执行时，避免淘汰锁定位为 1 的页面，从而减少缺页发生的次数。

6. 设某 UNIX 系统，文件系统的每个 inode 包含直接索引项 12 个和一、二、三级间接索引项各 1 个，物理块大小为 1KB，每个索引项占 4B，每个 inode 占 128B。存在一个大小为 12600 字节的文件 demo.bin。试阅读如下代码并回答问题（9 分）：

```
1  #include<stdio.h>
2  #include<unistd.h>
3  #include<fcntl.h>
4  #include<sys/wait.h>
5
6  int main(void)
7  {
8      int fd, rd_count;
9      char buf[600];
10
11     fd = open("demo.bin", O_RDONLY);
12
13     if (fork() == 0)
14     {
15         sleep(1);
16         fd = dup(fd);
17         rd_count = read(fd, buf,  600);
18         printf("%d: %d\n", fd, rd_count);
19     } else {
20         lseek(fd, 12000, 0); // 移动文件指针至第12000个字节处
21         rd_count = read(fd, buf, 200);
22         printf("%d: %d\n", fd, rd_count);
23         wait(NULL);
24     }
25
26     return 0;
27 }
```

（一） 该文件系统中 inode 块中文件物理结构信息占多少字节？单个文件的理论最大尺寸是多少（**KB 为单位，给出计算公式即可，不必最终运算结果**）？

（二） 第 11 行中 fd 值一般情况下是多少？试说明这个整型数的含义？在不修改上述代码的情况下，此代码运行时该值有没有可能为 0？若有可能，请给出实施方法。

（三） 第 16 行 dup 为文件描述符复制，为何需要这样的系统调用？该行代码顺利运行完成后，相应的**系统已打开文件表项**中**引用计数器**的值是多少？

（四） 假设文件系统有缓存机制，上述代码运行前无进程访问过 demo.bin 文件，代码运行过程中已读入的数据块至少会驻留内存直至代码运行结束，试分析第 17 行代码执行过程中逻辑块到物理块的转换过程，实际会读入多少个数据块？

（五） 上述代码执行完成后的输出内容是什么？（假设代码开始运行时，用户已打开文件**表项数量为 3，编号从 0 开始**）

参考答案：

（一） 即 inode 中多重索引表所占空间，为 15*4=**60 字节**，文件最大尺寸，**（12＋256＋256^2＋256^3）* 1KB**

（二） 一般情况为 **3**；用户已打开文件表项索引号；可以，**先 close(0)，再 exec 调用该代码**

（三） 主要用于**输入输出重定向**，将文件或管道描述符拷贝至原输入或输出设备文件描述符；**引用计数器为 3**（fork 增加一次，dup 增加一次）

（四） 17 行代码需要读入 3 个物理块，但由于 21 行代码执行在前，已读入其中的 1 个物理块（被缓存），故 17 行代码**实际读入 2 个物理块**

（五）3: 200

4: 400

7. 设有一个包含 200 个柱面(编号 0 - 199)的磁盘，磁盘移动臂刚处理完 100 号柱面的请求，正向柱面号大的方向移动。接着依次到来磁盘访问请求（柱面号）：186、27、129、110、147、41、46、10、120。试回答如下问题（4 分）：

（一） 试分别用先来先服务、最短查找时间优先、电梯调度和扫描等四种移动臂调度算法，给出完成所有访问请求的顺序，并计算各算法中移动臂经历的总柱面数。

（二） 若有多个磁盘请求在同一柱面的多个不同扇区上，应采用什么措施提高磁盘访问的效率？

参考答案：

（一）先来先服务：100-**186-27-129-110-147-41-46-10-120，660**

最短查找时间优先：100-**110-120-129-147-186-46-41-27-10，262**

电梯调度：100-**110-120-129-147-186-46-41-27-10，262**

扫描：100-**110-120-129-147-186-199-46-41-27-10，288**

（二）**归并同一柱面的请求**，并对同一柱面请求按**循环排序**的原则（如从小号扇区到大号扇区，或者反之）依次完成请求

得分	

二、编程题（8）

1. 试用管程实现满足如下要求的读者写者问题：1) 允许多个读进程同时读取文件内容，但同时读的进程数不得大于 K 个；2) 任何时候只允许一个写进程修改文件内容；3) 写进程完成前不允许读进程读文件；4) 等所有已经开始读的进程完成后才允许写进程写。(**定义管程** Monitor，包括声明若干条件变量 condition，以及条件变量上的 wait 和 signal 操作用于实现相关管程函数；**定义读、写进程**，能正确调用前述管程函数来实现上述同步要求)

参考答案：

M 管程定义分析：

start_read():
读申请时，两种等待情形：1）读请求时，写请求尚未完成；2）读请求时正在读的已达到 K 个。

end_read():
唤醒有两种情形：1）还有正在等的读请求，则唤醒等待的读请求（对应上面讨论的情形 1）；2）没有正在等待的读请求，则唤醒等待的写请求。**（读者优先）**

start_write():
对于写进程，一种等待情形：有读进程没有完成

end_write():
优先唤醒等待的读请求，没有则唤醒写请求

根据以上分析，至少需要两个条件变量（也可三个条件变量）用于等待，若有读写共用的条件变量，则需要用 while 判断，并配合三个计数器，分别用于记录正在读的进程、等待的读进程和等待的写进程

读、写进程定义

reader:	writer:
M.start_read()	M.start_write()
读	写
M.end_read()	M.end_write()