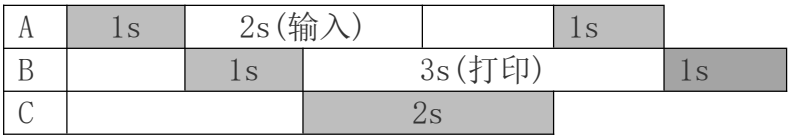
一、

（1）答：采用多道程序设计技术的系统，需要通道（或 DMA）和中断这两个辅助技术的支持，才能够达到多道程序设计的目标。

（2）答：多道程序设计技术主要应用在（脱机）批处理系统中，而分时技术主要应用在分时操作系统中；分时技术需要将 CPU 的时间分割为较小的时间片，并通过调度将时间片分配给系统中的进程，而多道程序设计技术则无需对 CPU 时间进行分割。从进程调度的观点来看，采用多道程序设计技术的系统，施行的是不可抢占调度，因为在这类系统中，系统无法中断正在使用 CPU 的程序的执行，而将 CPU 分配给另一道程序使用。

（3）深色表示是 CPU 占用时间



二、（1）答：P1 中的临界区为：

```
x = x + 1 ;
```

以及

```
y := 0 ;
print (y) ;
```

P2 中的临界区为：

```
y = y*y ;
```

以及

```
if( x<0 )
```

（2）答：定义两个信号灯：

mutex1，用于对 x 共享变量的访问互斥，初值为 1；

mutex2，用于对 y 共享变量的访问互斥，初值为 1；

程序描述：

```
Cobegin
P1;P2;
Coend
```

P1	P2

...
P(mutex1)
x = x + 1 ;
V(mutex1)
...
m = m*3;
...
P(mutex2)
y := 0 ;
print (y) ;
V(mutex2)
...
	...
	P(mutex2)
	y = y*y ;
	V(mutex2)
	...
	P(mutex1)
	if(x<0){
	V(mutex1)
	...}else{
	V(mutex1)
	...}
	...
	z = z + n ;
	...

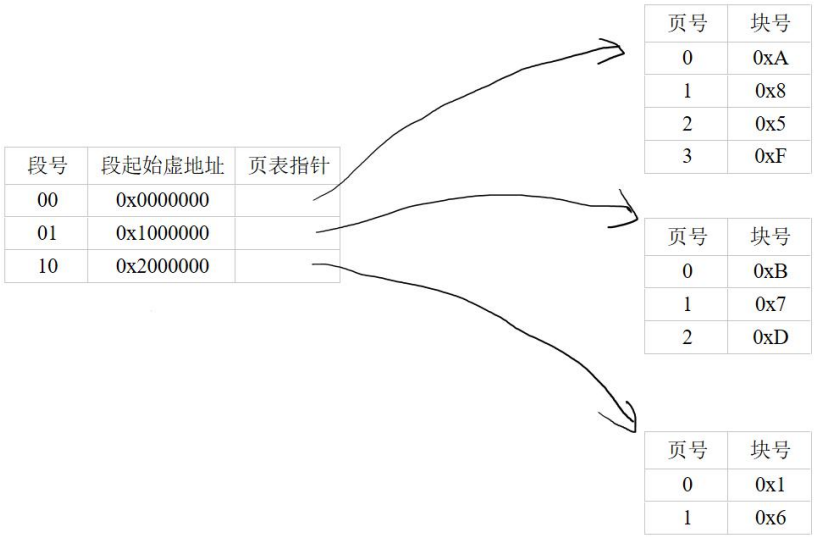
三、（1）答：该时刻系统处于安全状态。
（可能的）安全序列为：
①此时刻系统尚剩余 10-3-2-2=3 个资源，分配 2 个资源给 C 进程；
②C 进程获得所需资源，执行完毕，并最终释放 4 个资源。此时系统剩余 3-2+4=5 个资源；
③将 4 个资源分配给 A 进程 ；
④A 进程获得所需的资源，执行完毕，并最终释放 7 个资源。此时系统剩余 5-4+7=8 个资源；
⑤系统剩余的 8 个资源中的 7 个分配给 B 进程。
⑥B 进程完成执行，并释放所有资源。

以上，A、B、C 进程都执行完毕，系统未出现死锁。

（2）答：该时刻若进程 B 申请使用 1 个资源，系统能够允许 B 进程的资源申请，并将 1 个资源分配给 B。因为分配 1 个资源给 B 后，系统仍然存在以下安全路径：C→A→B
这意味着若系统批准 B 进程的资源申请后，仍然处于安全状态。所以，系统能够批准 B 的资源申请。

四、（1）
答：进程的实际虚拟地址空间为：3*2^24 = 3*16MB=48MB。

（2）



段号	段起始虚地址	页表指针
00	0x0000000	
01	0x1000000	
10	0x2000000	

页号	块号
0	0xA
1	0x8
2	0x5
3	0xF

页号	块号
0	0xB
1	0x7
2	0xD

页号	块号
0	0x1
1	0x6

（3）答：逻辑地址 0x10006AD 的第 24~25 位的值为 01，说明该逻辑地址位于数据段。因为页面大小为 1KB，说明页内位移的地址范围是 0~9 位（10 个位），所以 0x6AD 的页号 P=0x6AD>>10=0x1，W=0x6AD&0x3FF=0x2AD。查找数据段的页表可知，逻辑地址位于的块号为 P’=0x7,将 P’和 W 进行装配，可得物理地址为：0x1EAD。

五、（1）答：

采用先进先出替换算法，页号栈变化情况如下：

访问：	0	2	1	0	5	3	0	2
			1	1	5	3	0	2
		2	2	2	1	5	3	0
	0	0	0	0	2	1	5	3
缺页？	1	2	3		4	5	6	7

缺页中断次数：7 次

（2）答：

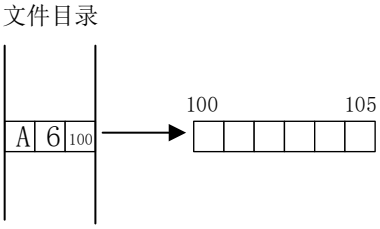
采用最久未使用置换算法，页号栈变化情况如下：

访问：	0	2	1	0	5	3	0	2
			1	0	5	3	0	2
		2	2	1	0	5	3	0
	0	0	0	2	1	0	5	3
缺页？	1	2	3		4	5		6

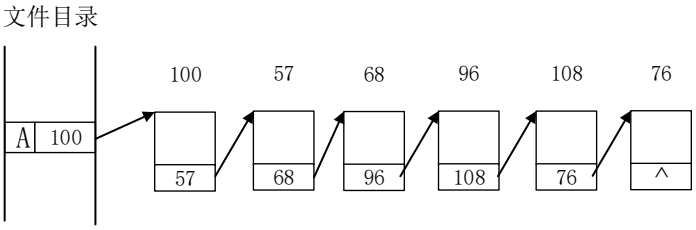
缺页中断次数：6 次

六、（1）答：

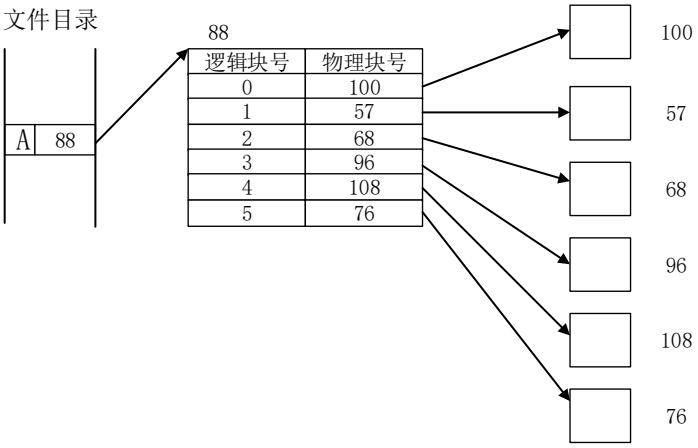
连续结构



串联结构



索引结构



（2）答：连续结构：1 次；串联结构：5 次；索引结构：2 次。

七、（1）答：单个磁盘块能够容纳的目录项为：（512-2）/51=10 个；每个目录文件最多占用 5 个磁盘块，则一个目录最多能够容纳的文件个数为：10*5=50 个。
（注：学生可能会考虑到一个目录需要预留一个目录项，存放父目录的目录项，如果是这种情况的话，则答案为 49 个，也是对的）。

（2）答：查找/home/user/os/os1.c 文件需要依次查找的子目录名为：在根目录的目录文件中查找 home;在/home 目录的目录文件中查找 user; 在/home/user 的目录文件中查找 os；以及在/home/user/os 的目录文件中查找 os1.c。一共 4 次查找过程。
假设每次查找目录都会在目录文件的第一个块中找到目录项(即读磁盘最少的情况），总共所需的读磁盘次数为：4 次。

假设每次查找目录都会在目录文件的最后一个块中找到目录项(即读磁盘最少的情况），总共所需的读磁盘次数为：4*5=20 次。

（3）答：最少的情况是直接索引，这种情况下会有 1 次读取磁盘块的动作，因为文件已被打开，其 i 节点已经位于内存；最多的情况是数据位于被二级间接索引的磁盘块中，这种情况下会有 3 次读取磁盘块的动作，分别对应：二级索引块、一级索引块，和数据块。

八、（1）答：定义信号灯：
empty，用于表示缓冲区是否为空，初值为 1；
data，用于表示缓冲区是否有数据，初值为 0；
cobegin
读卡机(); 进程();
coend

读卡机()
while（还有卡片）{
  P(empty)；
  读取卡片，并将数据送入缓冲区；
  V（data）；
}

进程()
While（工作未结束）{
  P(data)；
  根据缓冲区中的数据进行计算；
  V（empty）；
}

系统整体速度：
考虑一张卡片，它在读卡机上所花的时间为 1/x（秒）；在进程端，一张卡片所对应的数据需要花费的处理时间为 1/y（秒）。由读卡机和进程的同步关系可知，卡片的处理是顺序在两者间进行的，所以一张卡片在系统中所花费的时间为 1/x + 1/y（秒），系统的整体效率为：1/(1/x+1/y)= x*y/(x+y)（卡片/秒）

（2）答：定义信号灯：
empty1，用于表示缓冲区 1 是否为空，初值为 1；
data1，用于表示缓冲区 1 是否有数据，初值为 0；
empty2，用于表示缓冲区 2 是否为空，初值为 1；
data2，用于表示缓冲区 2 是否有数据，初值为 0；
cobegin
读卡机(); 进程();
coend

读卡机()
while（还有卡片）{
  P(empty1)；
  读取卡片，并将数据送入缓冲区；
  V（data1）；
  P(empty2)；
  读取卡片，并将数据送入缓冲区；
  V（data2）；
}

进程()
While（工作未结束）{
  P(data1)；
  根据缓冲区中的数据进行计算；
  V（empty1）；
  P(data2)；
  根据缓冲区中的数据进行计算；
  V（empty2）；
}

系统整体速度：
由读卡机和进程的同步关系可知，两者并行工作，系统的整体速度取决于较慢的部件。所以，系统的整体速度为 min(x,y) （卡片/秒）

九、（1）答：定义信号灯：
seats，对应 100 个座位，初值为 100；
register，对应前台，初值为 1；

cobegin
读者 1(); 读者 2();…; 读者 n();
coend

读者 i()
{
P(seats);
进入图书馆；
P(register)；
登记；
V(register)；

阅读；

P(register)；

注销；
V(register)；
V(seats)；
离开；
}

（2）答：
定义共享变量：
int readers=0;　　用于记录读者的数量，初值为 0

定义信号灯：
register，对应前台，初值为 1；
mutex，用于对 readers 访问的互斥，初值为 1；

cobegin
读者 1(); 读者 2();…; 读者 n();
coend

读者 i()
{
P(mutex)
if(readers >= 100){
    V(mutex)
    转身离开；
}else{
    readers++;
    V(mutex);
}

P(register)；
登记；
V(register)；

阅读；

P(register)；
注销；
V(register)；
P(mutex)
readers--;
V(mutex);

离开；

```
}
```