操作系统 2005 年试卷

一.

1 .操作系统的功能：控制应用程序的执行，并充当应用程序和计算机硬件之间的接口。

2 .临界区：是一段代码，在这段代码中进程将访问共享资源，当另外一个进程已经在这段代码中运行时，这个进程就不能在这段代码中执行。

3 .需要“挂起”状态。当主存中的所有进程都处于阻塞态时，操作系统可以把其中一个进程置于挂起态，并将它转移到磁盘，主存中释放的空间可以被调入的另一个进程使用。

4 .执行系统调用时需要改变进程的执行模式。系统调用需要内核的参与，执行内核程序要在内核模式下，因此需要模式切换。

5 .切换的开销：用户级线程的管理工作完全由应用程序完成，即在用户模式下进行，线程的切换不涉及到模式的切换。

内核级线程的管理工作由内核完成，即需要在内核模式下进行，线程的切换需要进行模式切换。

并行性：在用户级线程下，内核没有意识到线程的存在，它以进程为单位进行调度，因此，任何时候，一个进程中最多只有一个线程在运行。

在内核级线程下，线程的调度完全由内核完成，因此，在同一时刻，一个进程中不同线程可以并行执行。

6 .虚拟内存有效性的依据是：局部性原理。

局部性原理描述了一个进程中程序和数据的集簇倾向。

7 .页越小，内部碎片的总量越少，但每个进程需要的页的数目就越多，这就意味着更大的页表。如果页尺寸非常小，页错误率比较低，当页尺寸增加时，页错误率开始增长，但当页尺寸接近整个进程的大小时，页错误率开始下降，当一个页包含整个进程时，不会发生页错误。

8 .^_^

P1:C1=20,T1=100,D1=100

P2:C2=30,T2=145,D2=145

P3:C3=68,T3=150,D3=150

总使用率为:20/100+30/145+68/150=0.86

$$3\times(2^{1/3}-1)=0.78<0.86$$

具体调度我就不写了^_^.....

9 .RAID1 可靠性更好，因为它包含有磁盘数据的一个复本。

RAID1 写磁盘的速度更快，因为它可以同时并行的写数据磁盘和备份磁盘。而 RAID5 则需要更新校验位，要先读取旧的校验信息，更新后再写入新的校验信息。

10.饥饿不是死锁的充分条件；饥饿是死锁的的必要条件。

死锁的进程不一定处于阻塞状态。

二.

```
void lockAcquire(Lock *lock)
{
    int oldValue;
    intrSet(IntrOff,&oldValue);
```

```
    while(lock->lock==1)
        {
            intrSet(IntrOn,NULL);
            block(lock);
            intrSet(IntrOff,&oldValue);
        }
    lock->lock = 1;
    intrSet(IntrOn,NULL);
}

void lockRelease(Lock *lock)
{
    int oldValue;
    intrSet(IntrOff,&oldValue);
    if(!isEmpty(lock))
            unblock(lock);
    lock->lock = 0;
    intrSet(IntrOn,NULL);
}

void initLock(Lock *lock ,int v)
{
    int oldValue;
    intrSet(IntrOff,&oldValue);
    if(v==0)
        lock->lock = 0;
    else
        lock->lock = 1;
    intrSet(IntrOn,NULL);
}
```

三.

1．While 语句保证了在比较票号之前没有人正在拿票，即当对 number[j]进行写操作时，不允许其他人对 number[j] 进行读操作，因此，这里实现了对 number[j]的读-写互斥。

2．该算法实现了两个互斥：

A.临界资源的使用互斥

当 i 在临界区时，j 想进入临界区，此时必有(number[i],i)<(number[j],j).

原因如下：

1.若 j 在 i 进入临界区后才开始拿票准备进入临界区，则根据

Number[j] = 1+getmax(number[],n);

可知，number[j]一定大于 number[i]，因此 i 请求进入临界区无法得到满足

2.若 j 在 i 进入临界区前已拿到票，则 i 能进入临界区说明了(number[i],i)<(number[j],j)

B.number[j]的读写互斥

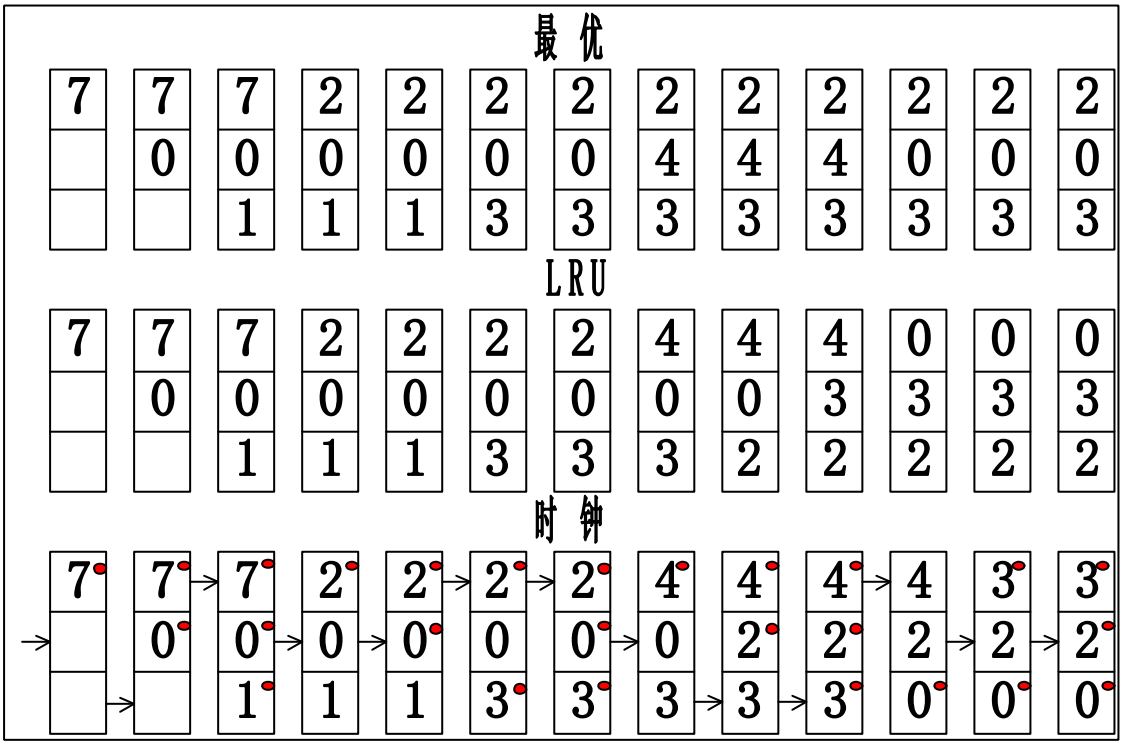
Choosing[j]保证了当对 number[j]写时，不允许对其读。

四.

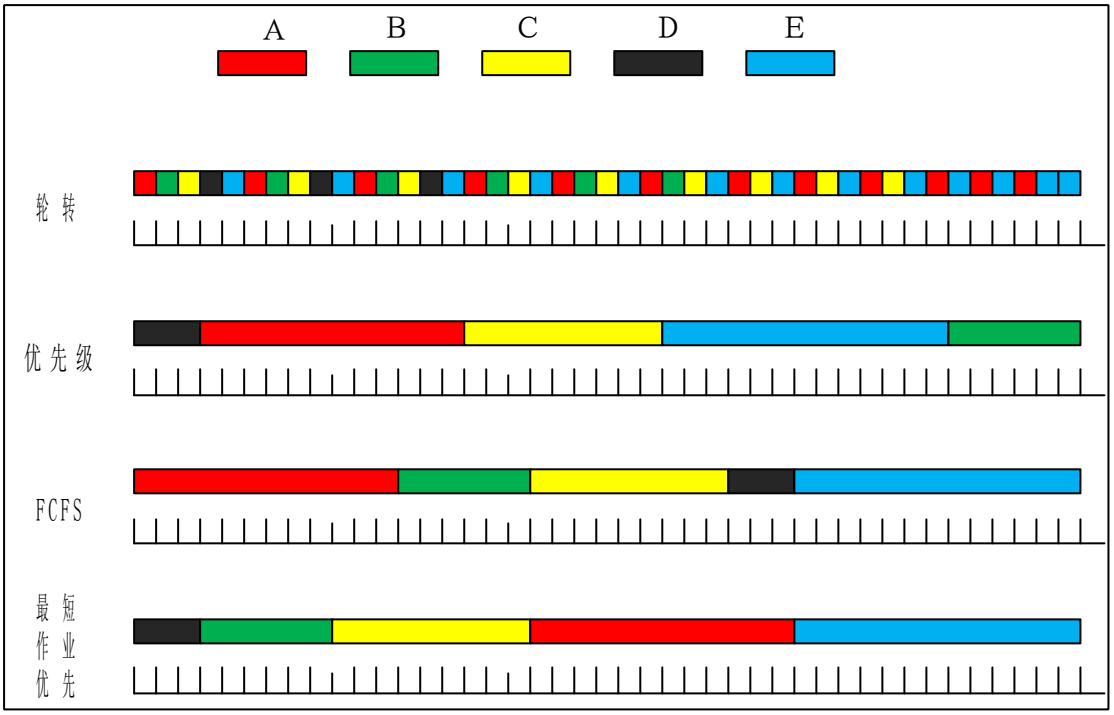
```
int Translate(int virtAddr,int *physAddr)
{
	int slot;
	slot = virtAddr/PageSize;
	if(slot>=pageTableSize)//Space is out of bound
	{
		physAddr = NULL;
		return 1;
	}
	*physAddr = pageTable[slot].physicalPage*PageSize+virAddr%PageSize;
	return 0;
}

int ReadMem(int addr,int size,int *value)
{
	if(*value+size-1>NumPhysPage*PageSize)
		return 1;
	int realAddr;
	int status = Translate(addr,&realAddr);
	if(status)
		return 1;
	for(int i=0;i<size;++i)
	   mainMemory[*value+i] = mainMemory[realAddr+i];
	return 0;
}
```

五



六.



周转时间计算如下：

1.时间片轮转

Ta＝41　Tb＝25　Tc＝35　Td＝14　Te＝43

2.优先级

Ta＝15　Tb＝43　Tc＝24　Td＝3　Te＝37

3.FCFS

Ta＝12　Tb＝18　Tc＝27　Td＝30　Te＝43

4.最短作业优先

Ta＝30　Tb＝9　Tc＝18　Td＝3　Te＝43

七.

Process 1
Outgoing channels
3 sent 1,2,3,4
Incoming channels
2 received 1,2,3,4,5 stored 6,7

Process 2
Outgoing channels
1 sent 1,2,3,4,5,6,7
3 sent 1,2,3,4,5,6,7
4 sent 1,2,3,4,5,6,7
Incoming channels
3 received... stored 1,2,3

Process 3
Outgoing channels
2 sent 1,2,3
4 sent 1,2,3
Incoming channels
1 received 1,2,3,4
2 received 1,2,3,4,5,6 stored 7

Process 4
Outgoing channels
Incoming channels
2 received... stored 1,2,3,4,5,6,7
3 received 1,2,3

八.

直接地址能表示的文件大小：10*4K=40K

一级间接地址能表示的文件大小：4K/4*4K=4M

二级间接地址能表示的文件大小为：1K*1K*4K=4G

三级间接地址能表示的文件大小为：1K*1K*1K*4k=4T

现在，文件的大小为 4G+8M+40K,因此需要用到三级间接地址，且用之表示 4M大小的文件。

总的间接地址所需的附加磁盘存储空间为：

4K+1K*4K+4K+3*4K=20K+4M