典型习题

一、论述

1、线程是执行流的抽象，为了在进程地址空间中运行多个线程，线程应该配备什么样的结构才能互不干扰？

要点：引入线程概念之后，一个进程中可以运行多个线程。每个线程从进程程序中定义的一个过程（procedure）入口开始执行。进程中的多个线程共享进程地址空间中的代码区和数据区，因此多个线程可能访问同一数据，多个线程也可能执行同一 procedure。当多个线程访问同一数据时要注意对数据的互斥保护，当多个线程执行同一 procedure 时，必须为每一个线程配备各自的**栈（stack）结构**和**线程控制块结构（TCB）**。栈结构也称为活动记录（active record），记录执行一个过程时，所有输入、输出参数和过程内部局部变量的值。由于每个线程都有自己的栈，因此执行同一 procedure 的多个线程不会出现**数据**的冲突。又由于每个线程具有自己的 TCB，而 TCB 中又记录着下一条指令的地址以及上次切换时处理器的寄存器状态，因此多个线程之间也不会出现**指令**的冲突。综合这两条，线程应配备线程栈和线程控制块，线程才不会相互干扰。当然对共享数据的互斥保护必须由程序员来进行设计。

2、从调度、内存管理的角度分析，操作系统如何实现多道程序设计？

要点：多道程序设计是指在单处理器计算机上同时运行多个应用程序，这些应用程序在宏观上是同时执行的，在微观上是交叠执行的。为了实现多道程序设计，操作系统在调度和内存管理方面必须采取一定的设计策略。在调度管理方面，首先对进程的运行状态进行划分，将进程分为运行、就绪和阻塞等基本状态，并根据进程的状态决定是否要对它进行调度；其次对调度模式进行划分，分为抢占式和不可抢占式；然后才去相应的调度算法，如 FIFO，基于优先级等。通过处理器的调度算法实现了多道进程在宏观上的同时运行。

由于多个进程同时运行，它们共享着同一内存空间，因此必须对内存进行合理的划分，使得进程不能相互冲突。比如，当分页式内存管理时，每个进程都有自己的页表，页表中的页表项对应不同的物理页面，通过页表把不同进程映射到不同物理页面分布，实现了进程的物理地址空间隔离。另一方面，当多个进程具有共享数据（如 shared memory，file）和共享代码（如共享库）的需求时，又通过页表把共享的数据和代码映射到同一物理地址空间，实现了进程间的共享。总之，隔离与共享是一对矛盾，都可以通过内存管理来实现。

3、谈谈如何用页表实现进程私有空间的隔离以及公共区的共享？

要点：采用分页式内存管理时，每个进程都有自己的页表，页表中的页表项对应不同的物理页面，通过页表把不同进程映射到不同物理页面分布，实现了进程的物理地址空间隔离。另一方面，当多个进程具有共享数据（如 shared memory，file）和共享代码（如共享库）的需求时，又通过页表把共享的数据和代码映射到同一物理地址空间，实现了进程间的共享。总之，隔离与共享是一对矛盾，都可以通过内存管理来实现。

4、从存储位置和生命周期的角度分析文件和打开文件的关系？

要点：文件存储在永久性存储介质中，如磁盘、磁带、光盘等。其生命周期跨越计算机的开关机周期；而打开文件是为了实现进程对文件的操纵，操作系统在内存中为文件建立的一系列数据结构，具体包括 v-node 结构、系统打开文件表、进程打开文件表等。打开文件数据

结构存储在内存中，它随着文件被首次打开而建立，随着文件被最后关闭而消亡，因此它们的生命周期不超过进程的生命周期。

二、分析与计算

1、有三个进程 P1、P2 和 P3。P1 向盘子放黑子或白子，P2 从盘中拣黑子，P3 从盘中拣白子。P1、P2 和 P3 不能同时操作，而且当盘子为空时 P1 才能放入。试写出三个并发进程能正确执行的程序。（用 Put、TakeBlack 和 TakeWhite 分别表示放入、拣黑子、拣白子操作）

解：

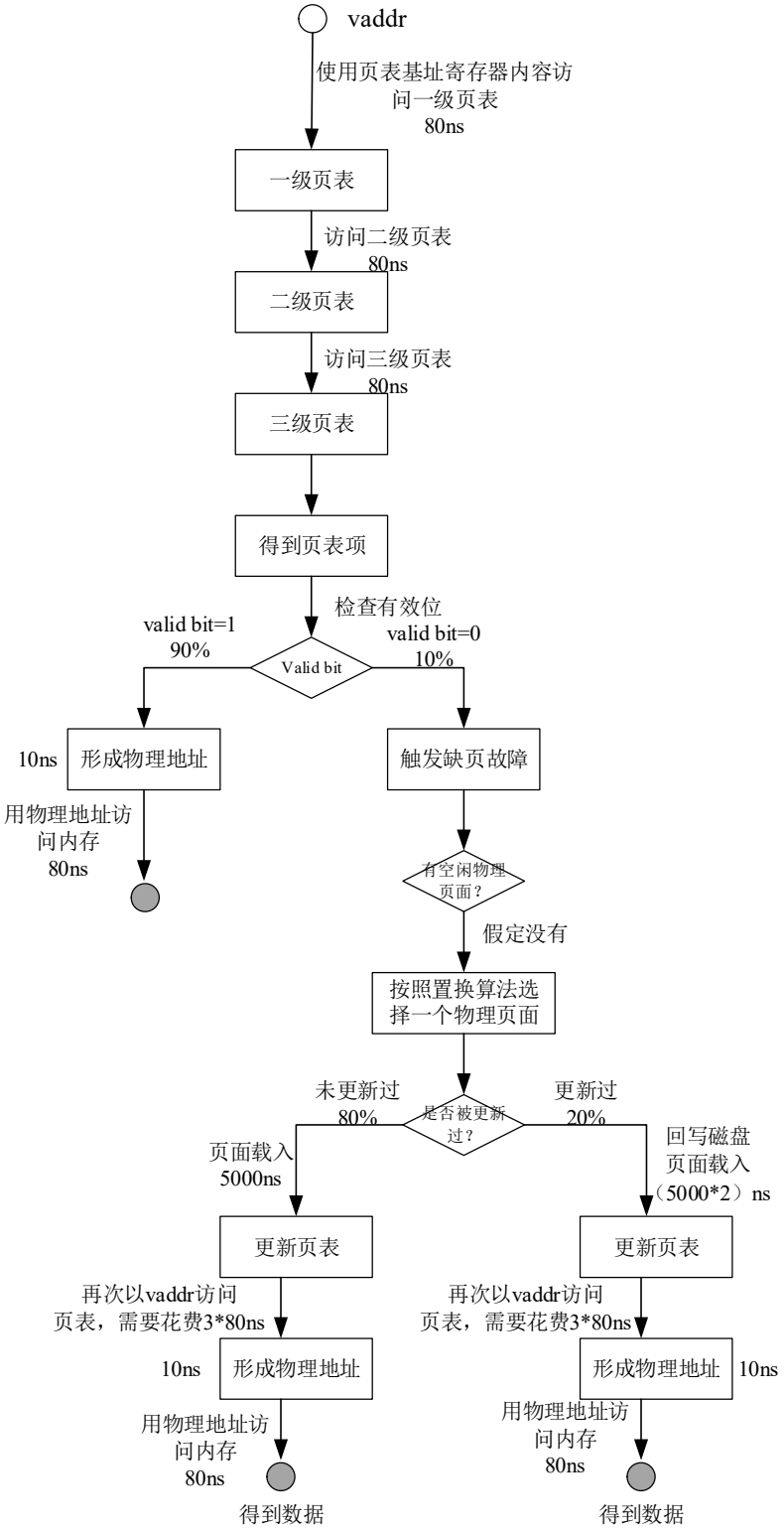
Semaphore empty:=1;　　//信号量，表示盘子是否为空

Semaphore back:=0, white:=0;　　//信号量，表示盘子中有黑子或白子

P1	P2	P3

while (1){
P(empty);
put(black); **V**(black);
□
put(white); **V**(white);
}
	while(1){
	P(black);
	TakeBlack();
	V(empty);
	}
		while(1){
		P(white);
		TakeWhite();
		V(empty);
		}

2、考虑这样一个分页系统，该系统在内存中存放了三级页表，不考虑 TLB。如果内存访问时间为 80ns，页面交换时间需要 5000ns，物理地址计算需要 10ns。假设有 20%的页面被更改，缺页率是 10%。求以虚拟地址访问到一个数据项的平均时间。



平均时间为：

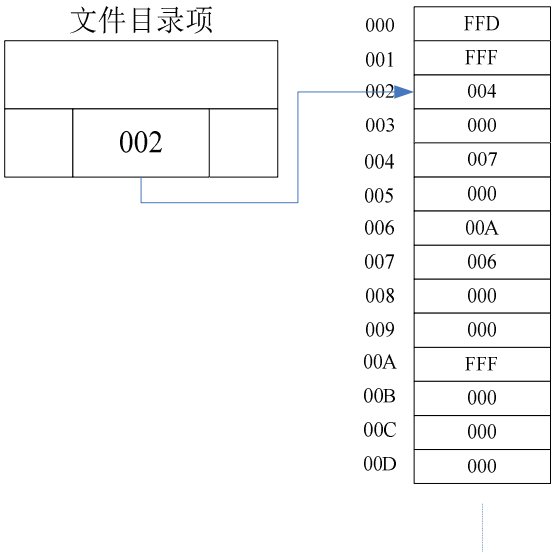
80*3+90%*(10+80)+10%*(80%*(5000+240+10+80)+20%*(5000*2+3*80+10+80))

=240+81+10%*(80%*5330+20%*10330)

=240+81+10%*(4264+2066)=240+81+633=954ns

3、如图所示是一个 MS-DOS 文件的 FAT 表。假设该磁盘大小为 50GB，盘簇大小为 1KB，盘

簇号为 4 字节。



（1） 求 FAT 表的大小（单位：字节）；

（2） 指出该文件所占据的盘簇序列；

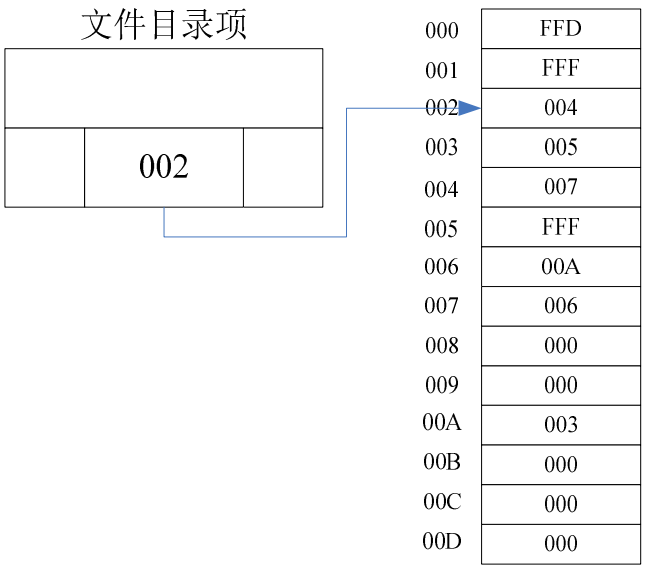
（3）假定在该文件末尾进行了写操作，使得它的大小扩展了 2 个盘簇，试在上述 FAT 表的基础上，为该文件分配盘空间，即写出分配后的 FAT 表。

解：

（1）50GB 磁盘共有 50GB/1KB=50M 个磁盘块，每个磁盘块号为 4 字节，而有多少个磁盘块，FAT 表里就有多少个表项，因此 FAT 大小为 50M*4B=200MB 大小。

（2）该文件所占据的盘簇序列为：002、004、007、006、00A。

（3）假定从第一个空白区进行分配，那么 FAT 表中 00A 号对应的表项内容应当为 003,003 号对应的表项内容应当为 005,005 号对应的表项内容为 FFF，其它表项内容保持不变，即



4、桌子上有一只盘子，最多可以容纳 1 个水果。爸爸一次只能向盘子中放入一个苹果，妈妈一次只能向盘子中放入一个桔子。女儿专等吃盘中的苹果，儿子专等吃盘子里的桔子。(用 PA，PO，TA，TO 分别表示向盘中“放苹果”，“放桔子”，从盘中“取苹果”和“取桔子”的操作)。试用 P、V 操作写出能同步的程序。

解：Sem SA:=0, SO:=0;

Sem Mutex:=1;

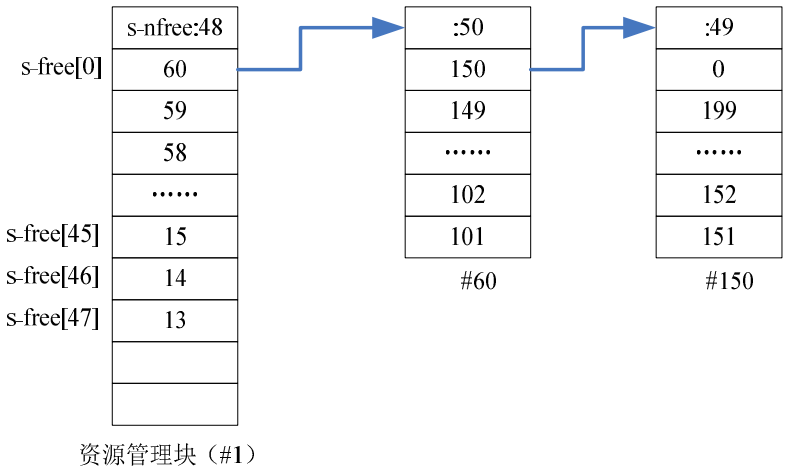
Parent	Mother	Son	Daught
P(Mutex); PA; P(SA);	P(Mutex); PO; P(SO);	V(SA); TA; V(Mutex);	V(SO); TO; V(Mutex);

5、Unix 采用成组链接法对磁盘空间进行管理，每一组为 50 个磁盘块（第一组为 49 个块）。某时刻 t0 磁盘空间的成组链表的状态如图所示。试计算：

（1）t0 时空闲磁盘块的个数；

（2）t0 时哪几个块用来存放空闲块号？实际用来存放空闲块号的磁盘块有几个？

（3）进程 P 依次释放了四个块：7、10、8、6 后，成组链表的状态（画图描述）。

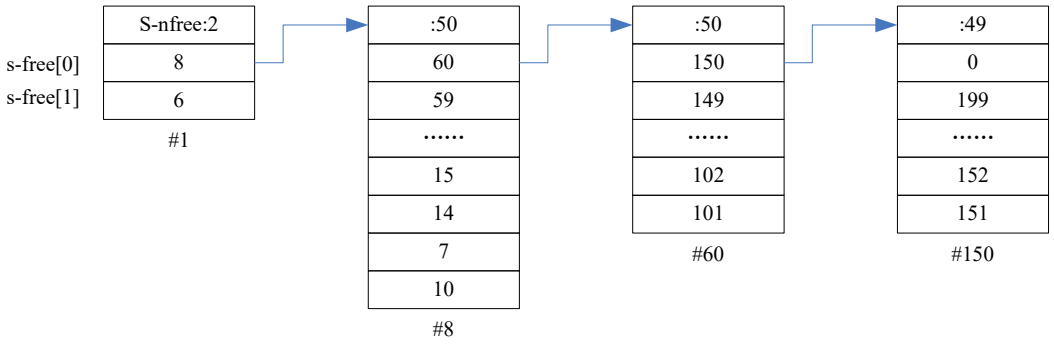


解：

（1）t0 时刻空闲磁盘块的个数为：49+50+48=147 块。

（2）t0 时刻，资源管理块即 1#块、60#块和 150#块用来存放空闲块号。实际用来存放空闲块号的只有 1#块，因为 60#和 150#块本身是空闲块，只是用来暂时存放空闲块号而已，可以被分配出去。因此实际用来存放空闲块号的只有 1 个块，即 1#块。

（3）进程 P 依次释放了四个块 7、10、8、6 后，成组链表的状态为：


S-nfree:2
s-free[0]
8
s-free[1]
6
#1
:50
60
59
……
15
14
7
10
#8
:50
150
149
……
102
101
#60
:49
0
199
……
152
151
#150

注：没有考虑块号的排序。