填空题

1、Linux 操作系统按照事件来源和实现手段将中断分为（ 硬中断 ）、（ 软中断 ）。
2、系统调用是通过（ 中断 ）来实现的；发生系统调用，处理器的状态常从目态变为管态。
3、在 Linux 系统中，创建进程的原语是（ fork ）。
4、进程的基本三状态模型并不足够描述进程的真实的情况，进程的五状态模型增加了两个状态，包括（ 新建状态 ）和（ 终止状态 ）。
5、系统中进程存在的唯一标志是（进程控制块 PCB）。
6、进程上下文包括了进程本身和运行环境，是对进程执行活动全过程的静态描述。进程上下文分成三个部分：（ 用户级上下文（进程的用户地址空间内容）)、（ 寄存器级上下文（硬件寄存器内容） ）和（ 系统级上下文（与该进程相关的核心数据结构）)。
7、进程调度方式通常有（ 抢占 ）和（ 非抢占 ）两种方式。
8、若信号量 S 的初值定义为 10，则对 S 调用执行了 16 次 P 操作和 15 次 V 操作后，S 的值应该为（ 9)。

简答题

1、请简单叙述进程三态模型中的进程状态转化情况。
答：

- 就绪态→运行态：当调度程序选择一个新的进程运行时，进程会由就绪态切换到运行态；
- 运行态→就绪态：当运行进程用完了获得的时间片时，进程就会被中断，由运行态切换到就绪态，或是因为一高优先级进程处于就绪状态，正在运行的低优先级进程会被中断而由运行态切换到就绪态；
- 运行态→等待态：以下几种情况会导致进程会由运行态切换到等待态，例如当一进程必须等待时，或是操作系统尚未完成服务，进程对一资源的访问尚不能进行时，还有初始化 I/O 且必须等待结果时，在进程间通信时，进程等待另一进程提供输入时等；
- 等待态→就绪态：当进程所等待的事件发生时，例如资源申请获得满足时，或是等待的数据或信号到来时，进程就可能由等待态切换到就绪态。

2、进程创建来源于以下事件：提交一个批处理作业；在终端上交互式的登录；操作系统创建一个服务进程；进程孵化新进程；等等。请描述进程的创建过程。
答：

① 系统在进程表中增加一项，并从 PCB 池中取一个空白 PCB；
② 为新进程的进程映像分配地址空间。传递环境变量，构造共享地址空间；
③ 为新进程分配资源，除内存空间外，还有其他各种资源；
④ 查找辅存，找到进程正文段并装到正文区；
⑤ 初始化进程控制块，为新进程分配进程标识符，初始化 PSW；
⑥ 加入就绪进程队列，将进程投入运行；
⑦ 通知操作系统的某些模块，如记账程序、性能监控程序。

3、请简述时间片轮转调度算法的工作流程和确定时间片大小需要考虑的因素。
答：

1、时间片轮转调度算法的工作流程：

- 系统将所有的就绪进程按先来先服务的原则排成一个队列，每次调度时把 CPU 分配给队首进程，并令其执行一个时间片。
- 当执行的时间片用完时，由系统中的定时器发出时钟中断请求，调度程序停止该进程的执行，并将它送到就绪队列的末尾，等待下一次执行。
- 进行进程切换，把处理器分配给就绪队列中新的队首进程。

2、时间片大小的确定要从进程个数、切换开销、系统效率和响应时间等方面考虑：

- 时间片取值太小，多数进程不能在一个时间片内运行完毕，切换就会频繁，开销显著增大，从系统效率来看，时间片取大一点好。
- 时间片取值太大，随着就绪队列里进程数目增加，轮转一次的总时间增大，对进程的响应速度放慢了。为满足响应时间要求，要么限制就绪队列中进程数量，要么采用动态时间片法，根据负载状况及时调整时间片的大小。

4、有两个优先级相同的并发运行的进程 P1 和 P2，各自执行的操作如下，信号量 S1 和 S2 初值均为 0，x、y 和 z 的初值为 0。

Cobegin	

P1：
begin
y:=0;
y:=y+4;
V(S1);
z:=y+3;
P(S2);
y:=z+y
end
	P2：
	begin
	x:=2;
	x:=x+6;
	P(S1);
	x:=x+y;
	V(S2);
	z:=z+x;
	end

Coend	

试问 P1、P2 并发执行后，x、y、z 的值有几种可能，各为多少？

答：

1：x=12，y=11，z=19。

2：x=12，y=23，z=19。

3：x=12，y=11，z=7。

5、为什么说最高响应比优先作业调度算法是对先来先服务以及短作业优先这两种调度算法的折中？

答：

先来先服务的作业调度算法，重点考虑的是作业在后备作业队列里的等待时间，因此对短作业不利；短作业优先的调度算法，重点考虑的是作业所需的 CPU 时间，因此对长作业不利。最高响应比优先作业调度算法，总是在需要调度时，考虑作业已经等待的时间和所需运行时间之比，即：

1 +（作业已等待时间 / 作业所需 CPU 时间）

比值的分母是一个不变的量。随着时间的推移，一个作业的“已等待时间”会不断发生变化，也就是分子在不断地变化。

显然，短作业比较容易获得较高的响应比。这是因为它的分母较小，只要稍加等待，整个比值就会很快上升。另一方面，长作业的分母虽然很大，但随着它等待时间的增加，比值也会

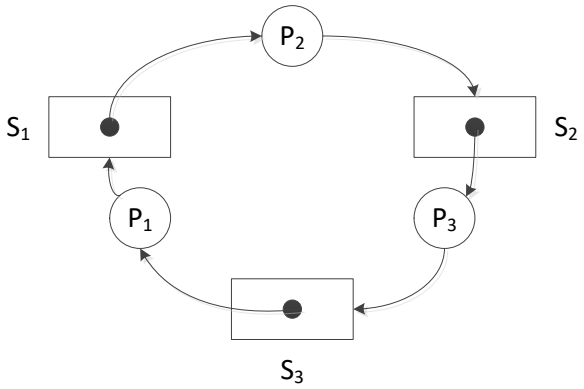
逐渐上升，从而获得较高的响应比。
可见最高响应比优先作业调度算法，既照顾到了短作业的利益，也照顾到了长作业的利益，是对先来先服务以及短作业优先这两种调度算法的一种折中。

6、请对比操作系统中“死锁”和“饥饿”问题。
答：

- 死锁是因进程竞争资源，但系统拥有资源的数量有限，或并发进程推进的顺序不当而造成的一种永远等待资源的僵局。
- 饥饿是指每个资源占用者都在有限时间内释放占用的资源，但申请进程仍然长时间得不到资源的现象，常常是策略不公平的体现。

7、一个计算机有 6 台设备 X，有 n 个进程竞争使用，每个进程最多需要两台。n 最多为多少时，系统不存在死锁的危险？
答：
由于每个进程最多需要两台设备 X，考虑极端情况：每个进程已经都申请了一台。那么只要还有一台空闲，就可以保证所有进程都可以完成。也就是说当有条件：n+1=6（即 n=5）时，系统就不存在死锁的危险。

8、3 个进程 P_1、P_2 和 P_3 并发工作。进程 P_1 需用资源 S_3 和 S_1；进程 P_2 需用资源 S_1 和 S_2；进程 P_3 需用资源 S_2 和 S_3。
（1）若对资源分配不加限制，会发生死锁情况，请画出发生死锁时，3 个进程和 3 个资源之间的进程资源分配图。
（2）为保证进程正确工作，应采用怎样的资源分配策略。
答：
（1）不加限制会出现死锁情况：



（2）可以采用的方法有多种，下面是几种可行的方法：

- 分配资源时，一次性分配该进程运行过程中所需的所有资源。破坏了死锁的必要条件之一“请求和保持条件”。
- 申请资源时，如果不能立即获得新的资源，则释放已经获得的资源。破坏死锁的必要条件之一“不可剥夺条件”。
- 对所有的资源进行编号，每个进程在申请资源时，严格按照资源编号递增的次序申请资源。这种方法是破坏了死锁的必要条件之一“环路等待条件”。

解答题

1、某系统有三个作业：

作业	到达时间	所需 CPU 时间
1	8.8	1.5
2	9.0	0.4
3	9.5	1.0

系统确定在它们全部到达后，开始采用响应比高者优先调度算法，并忽略系统调度时间。试问对它们的调度顺序是什么？各自的周转时间是多少？请写出计算过程，并填写下面表格。

答：

三个作业是在 9.5 时全部到达的。这时它们各自的响应比如下：

作业 1 的响应比 =（9.5 – 8.8）/ 1.5 = 0.46

作业 2 的响应比 =（9.5 – 9.0）/ 0.4 = 1.25

作业 3 的响应比 =（9.5 – 9.5）/ 1.0 = 0

因此，最先应该调度作业 2 运行，因为它的响应比最高。它运行了 0.4 后完成，这时的时间是 9.9。再计算作业 1 和 3 此时的响应比：

作业 1 的响应比 =（9.9 – 8.8）/ 1.5 = 0.73

作业 3 的响应比 =（9.9 – 9.5）/ 1.0 = 0.40

因此，第二个应该调度作业 1 运行，因为它的响应比最高。它运行了 1.5 后完成，这时的时间是 11.4。第三个调度的是作业 3，它运行了 1.0 后完成，这时的时间是 12.4。整个实施过程如下。

作业	到达时间	所需 CPU 时间	开始时间	完成时间	周转时间
1	8.8	1.5	9.9	11.4	2.6
2	9.0	0.4	9.5	9.9	0.9
3	9.5	1.0	11.4	12.4	2.9

作业的调度顺序是 2→1→3。各自的周转时间为：作业 1 为 0.9；作业 2 为 2.6；作业 3 为 2.9。

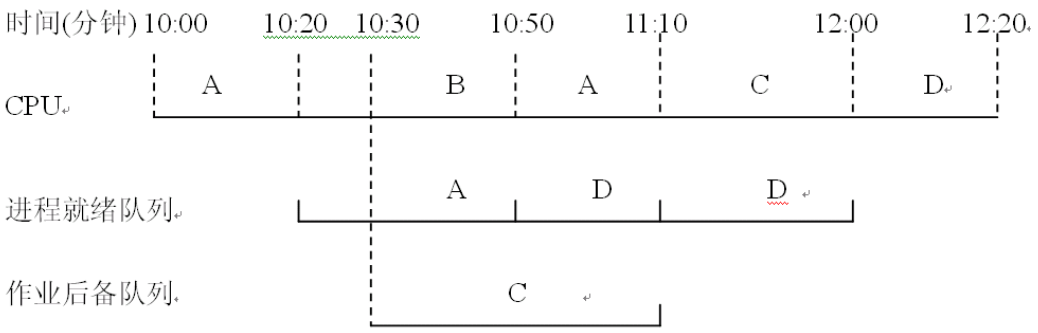
2、有一个具有两道作业的批处理系统，作业调度采用短作业优先的非抢式调度算法，进程调度采用以优先数为基础的抢占式调度算法，在下表所示的作业序列中，作业优先数即为进程优先数，优先数越小优先级越高。

作业	到达时间	所需 CPU 时间	优先数
A	10:00	40 分钟	5
B	10:20	30 分钟	3
C	10:30	50 分钟	4
D	10:50	20 分钟	6

列出所有作业进入内存时间及结束时间，并计算平均作业周转时间。

答：

(1)每个作业运行将经过两个阶段：作业调度(SJF 算法)和进程调度(优先数抢占式)。另外，批处理最多容纳 2 道作业，更多的作业将在后备队列等待。



a) 10:00，作业 A 到达并投入运行。
b) 10:20，作业 B 到达且优先权高于作业 A，故作业 B 投入运行而作业 A 在就绪队列等待。
c) 10:30，作业 C 到达，因内存中已有两道作业，故作业 C 进入作业后备队列等待。
d) 10:50，作业 B 运行结束，作业 D 到达，按短作业优先算法，作业 D 被装入内存进入就绪队列。而由于作业 A 的优先级高于作业 D，故作业 A 投入运行。
e) 11:10，作业 A 运行结束，作业 C 被调入内存，且作业 C 的优先级高于作业 D，故作业 C 投入运行。
f) 12:00，作业 C 运行结束，作业 D 投入运行。
g) 12:20，作业 D 运行结束。

各作业周转时间为：作业 A　70，作业 B　30，作业 C　90，作业 D　90。

(2) 平均作业周转时间为 70 分钟。

作业	进入内存时间	运行结束时间	作业周转时间	平均作业周转时间
A	10:00	11:10	70	70
B	10:20	10:50	30	
C	11:10	12:00	90	
D	10:50	12:20	90	

3、有一个垃圾分拣机器人系统，拥有两个机器手臂，可分别自动在垃圾箱里面分拣可回收易拉罐和塑料瓶。设分拣系统有二个进程 P1 和 P2，其中 P1 驱动左臂拣易拉罐；P2 驱动右臂拣塑料瓶。规定每个手臂每次只能拣一个物品；当一个手臂在拣时，不允许另一个手臂去拣；当一个手臂拣了一个物品后，必须让另一个手臂去拣。试用信号量和 P、V 操作实现两进程 P1 和 P2 能并发正确执行的程序。

答：

实质上是两个进程的同步问题，设信号量 S1 和 S2 分别表示可拣易拉罐和塑料瓶，不失一般性，若令先拣易拉罐。

```
var   S1,S2:semaphore;   S1:=1;S2:=0;
cobegin
{
process P1
   begin
      repeat
        P(S1);
```

```
        拣易拉罐
        V(S2);
        until false;
    end
process P2
    begin
        repeat
        P(S2);
        拣塑料瓶
        V(S1);
        until false;
    end
}
coend.
```

4、桌上有一只空盘子，允许存放一只水果。爸爸可向盘中放苹果和桔子，儿子专等着取盘中的桔子然后吃掉，女儿专等着取盘中的苹果然后吃掉。规定盘子一次只能放一只水果，盘子中水果没有被取走时，爸爸不可放新水果；盘子中没有水果时，女儿和儿子来取水果时将需等待。请用信号量和 P、V 原语实现爸爸、儿子、女儿 3 个并发进程的同步。

答：

设置 3 个信号量：

```
int S=1;//盘子是否为空，开始为空
int Sa=0;//  盘子是否有苹果
int Sb=0;//  盘子是否有桔子
Cobegin
Father()
{
While(1)
{
      P(S);
      水果放入盘中；
If(放入的是桔子) V(Sb);
Else V(Sa);
}
Son()
{
      While(1)
      {
      P(Sb);
      从盘中取出桔子；
      V(S);
      吃桔子；
      }
}
```

```
Daughter()
{
    While(1)
    {
    P(Sa);
    从盘中取出苹果；
    V(S);
    吃苹果；
    }
}
Coend
```

5、内存中有一组缓冲区被多个生产者进程、多个消费者进程共享使用，总共能存放 10 个数据，生产者进程把生成的数据放入缓冲区，消费者进程从缓冲区中取出数据使用。缓冲区满时生产者进程就停止将数据放入缓冲区，缓冲区空时消费者进程停止取数据。数据的存入和取出不能同时进行，试用信号量及 P、V 操作来实现该方案。

答：

```
semaphore mutex, empty, full;
mutex=1;        //互斥信号量
empty=10;           //生产者进程的同步信号量
full=0;         //消费者进程的同步信号量
    cobegin
  process Pi   //生产者进程
  {
      while (1) {
          生产数据 x;
          P(empty)      //看看是否还有空间可放
          P(mutex);     //互斥使用
          放入;
          V(full);      ///增 1(可能唤醒一个消费者)
          V(mutex);
      }
  }
  process Cj   //消费者进程
  {
      while (1) {
          P(full)       //看看是否有数据
          P(mutex);     //互斥使用
          取出;
          V(emtpy);     //增 1(可能唤醒一个生产者)
          V(mutex);
      }
  }
coend
```

6、假定系统有三个并发进程 read, move 和 print 共享缓冲器 B1 和 B2。进程 read 负责从输入设备上读信息，每读出一个记录后把它存放到缓冲器 B1 中。进程 move 从缓冲器 B1 中取出一记录，加工后存入缓冲器 B2。进程 print 将 B2 中的记录取出打印输出。缓冲器 B1 和 B2 每次只能存放一个记录。要求三个进程协调完成任务，使打印出来的与读入的记录的个数，次序完全一样。请用信号量和 P、V 操作，写出它们的并发程序。

答：

```
begin SR, SM1, SM2, SP:semaphore;
B1,B2:record;
SR:=1;SM1:=0;SM2:=1;SP:=0
cobegin
process read
X:record;
begin R: (接收来自输入设备上一个记录)
X:=接收的一个记录；
P(SR)；
B1:=X;
V(SM1);
goto R;
end;
Process move
Y:record;
begin
M:P(SM1);
Y:=B1;
V(SR)
加工 Y
P(SM2)；
B2:=Y;
V(SP);
goto M;
end;
Process print
Z:record;
begin
P:P(SP);
Z:=B2;
V(SM2)
打印 Z
goto P;
end;
coend;
end;
```

7、用银行家算法避免系统死锁：

进程	已占有资源数				最大需求数			
	A	B	C	D	A	B	C	D
P1	3	0	1	1	4	1	1	1
P2	0	1	0	0	0	2	1	2
P3	1	1	1	0	4	2	1	0
P4	1	1	0	1	1	1	1	1
P5	0	0	0	0	2	1	1	0

当前系统资源总量为 A 类 6 个、B 类 3 个、C 类个 4、D 类 2 个。

（1）系统是否安全？请分析说明理由。

（2）若进程B请求(0,0,1,0)，可否立即分配？请分析说明理由。

答：

(1)

由已知条件可得 Need 和 Avaiable 矩阵如下：

进程	分配矩阵				尚需矩阵(Need)				可用资源数向量(Avaiable)			
P1	3	0	1	1	1	1	0	0	1	0	2	0
P2	0	1	0	0	0	1	1	2				
P3	1	1	1	0	3	1	0	0				
P4	1	1	0	1	0	0	1	0				
P5	0	0	0	0	2	1	1	0				

利用银行家算法对此时刻的资源分配情况进行分析如下表：

进程	Work				Need				Allocation				Work+Allocation				Finish
P4	1	0	2	0	0	0	1	0	1	1	0	1	2	1	2	1	true
P1	2	1	2	1	1	1	0	0	3	0	1	1	5	1	3	2	true
P2	5	1	3	2	0	1	1	2	0	1	0	0	5	2	3	2	true
P3	5	2	3	2	3	1	0	0	1	1	1	0	6	3	4	2	true
P5	6	3	4	2	2	1	1	0	0	0	0	0	6	3	4	2	true

从上述分析可知，存在一个安全序列 D，A，B，C，E，（答案不唯一），故当前系统是否安全的。

（2）若进程 B 请求(0,0,1,0)，试分配并修改相应的数据结构，则系统状态变为：

进程	分配矩阵				尚需矩阵(Need)				可用资源数向量(Avaiable)			
P1	3	0	1	1	1	1	0	0	1	0	1	0
P2	0	1	1	0	0	1	0	2				
P3	1	1	1	0	3	1	0	0				
P4	1	1	0	1	0	0	1	0				
P5	0	0	0	0	2	1	1	0				

利用银行家算法对此时刻的资源分配情况进行分析如下表：

进程	Work				Need				Allocation				Work+Allocation				Finish
P4	1	0	1	0	0	0	1	0	1	1	0	1	2	1	1	1	true
P1	2	1	1	1	1	1	0	0	3	0	1	1	5	1	2	2	true
P2	5	1	2	2	0	1	0	2	0	1	1	0	5	2	3	2	true
P3	5	2	3	2	3	1	0	0	1	1	1	0	6	3	4	2	true

P5	6 3 4 2	2 1 1 0	0 0 0 0	6 3 4 2	true

从上述分析可知，存在安全序列 D，A，B，C，E，（答案不唯一）故系统仍是否安全的，因此可以立即分配。

8、假定系统中有五个进程{P0、P1、P2、P3、P4}和三种类型资源{A、B、C}，A、B、C 资源的总数量分别为 10、5、7。各进程的最大需求、T0 时刻资源分配情况如下所示。

	资源最大需求量			已分配资源量		
进程	A	B	C	A	B	C
P0	7	5	3	0	1	0
P1	3	2	2	2	0	0
P2	9	0	2	3	0	2
P3	2	2	2	2	1	1
P4	4	3	3	0	0	2

（1）T0 时刻是否安全？若安全，请说明理由，并给出一个可能的安全序列。若不安全，请说明理由。

（2）若接下来 P4 继续请求资源(3,2,1)，则系统是否允许并响应该请求？若允许，请说明理由，并给出一个可能的安全序列。若不允许，请说明理由。

答：

（1）T0 时刻是安全的。因为此时，系统中的剩余资源量为（3,3,2）。此时，可以满足 P1 或 P3 的全部剩余资源请求。假设先满足 P1 的请求，则 P1 运行结束后，将资源返还操作系统，则系统中的剩余资源量为(5,3,2)。此时，可以满足 P3 或 P4 的要求。假设接下来先满足 P3 的要求，则 P3 运行结束后，将资源返还操作系统，则系统中的剩余资源量为(7,4,3)。此时，将可以满足 P0 或 P2 或 P4 的任意一个的资源请求。无论分配给谁，都不会发生死锁。于是，安全序列为 P1、P3、（后面的进程顺序任意）。当然，还能形成其它安全序列——P1、P4、P3、（后面的进程顺序任意）；P3、P1、（后面的进程顺序任意）；P3、P4、P1、（后面的进程顺序任意）。

（2）系统可以允许该请求。因为当将 P4 所需资源分配给 P4 后，系统剩余资源为(0,1,1)。此时，剩余资源仅能满足 P3 的所有资源请求。假设将资源分配给 P3，则当 P3 运行结束后，将资源返还操作系统，则系统中的剩余资源量为(2,2,2)，可以满足 P1 或 P4 的剩余资源请求。于是，假设把资源分配给 P1，则当 P1 运行结束并归还资源后，系统剩余资源量为(4,2,2)；然后，再满足 P4，把资源分配给 P4，则当 P4 运行结束并归还资源后，系统剩余资源量为(7,4,5)；此时，将可以满足 P0 或 P2 或 P4 的任意一个的资源请求。无论分配给谁，都不会发生死锁。于是，安全序列为 P3、P1、P4、P0、P2 和 P3、P1、P4、P2、P0。当然，还能形成其它安全序列——P3、P4、P1、P0、P2 和 P3、P4、P1、P2、P0。