第二章作业

1.如果系统中有N个进程，那么运行进程最多几个，最少几个？就绪进程最多几个，最少几个？等待进程最多几个，最少几个？

运行进程：最多1个，最少0个。

就绪进程：最多N-1个,最少0个。

等待进程：最多N个，最少0个。

2.进程有无如下状态转换，为什么？

（1）等待—运行

（2）就绪—等待

本题说的是进程状态的转换，进程状态的转换没有等待态（阻塞态）-运行态和就绪态-等待态（阻塞态）。因为进程转化为阻塞态是进程主动进行的。进程转为就绪态为进程被动进行的，所以可以推断没有等待态（阻塞态）-运行态和就绪态-等待态（阻塞态）。

一个进程在创建后将处于就绪状态。每个进程在执行过程中，任意时刻当且仅当处于上述三种状态之一。同时，在一个进程执行过程中，它的状态将会发生改变。引起进程状态转换的具体原因如下：

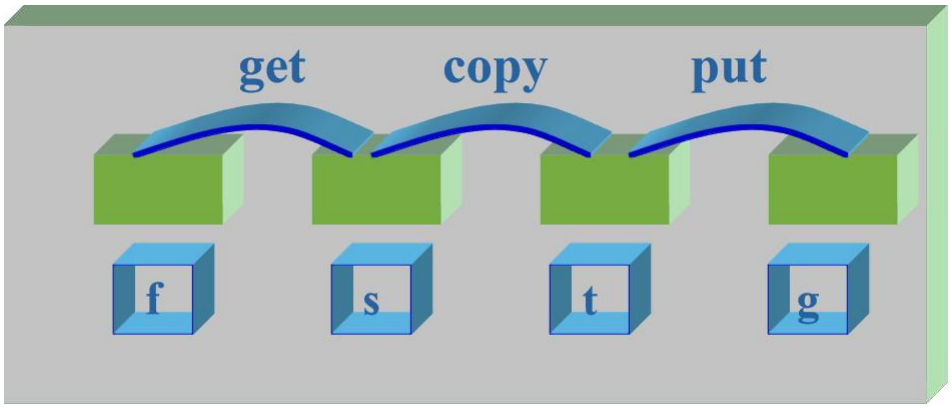
（1）运行态一一等待态：等待使用资源或某事件发生，如等待外设传输；等待人工干预。

（2）等待态一一就绪态：资源得到满足或某事件已经发生，如外设传输结束；人工干预完成。

（3）运行态一一就绪态：运行时间片到，或出现有更高优先权进程。

（4）就绪态一一运行态：CPU空闲时被调度选中一个就绪进程执行。

3.用P.V操作解决下图之同步问题



设置 8 个信号量 f_in = 1, f_out = 0, s_in = 1, s_out = 0, t_in = 1, t_out = 0, g_in = 1, g_out =1

假设 f 仓库原来是满的。

```
get:
while(1):
{
P(f_out);
P(s_in);
将数从f取出放入s;
V(f_in);
V(s_out);
}
```

```
copy:
while(1):
{
P(s_out);
P(t_in);
将数从s取出放入t;
V(s_in);
V(t_out);
}
```

```
put:
while(1):
{
P(t_out);
P(g_in);
将数从s取出放入t;
V(t_in);
V(g_out);
}
```

4、试从动态性、并发性、独立性和异步性上比较进程和程序

动态性：该特性是进程最基本的特性，可表现为由创建而产生、由调度而执行，因得不到资源而暂停执行，以及由撤销而消亡，因此进程有一定的生命周期。而程序只是一组有序指令的集合，是静态实体。

并发性：并发性是进程的重要特征，同时也是 OS 的重要特征。引入进程的目的正是为了使程序能和已建立进程的程序并发执行。而程序本身是不能并发执行的。

独立性：独立性是指进程实体是一个独立运行的基本单位，同时也是系统中独立获取资源和独立调度的基本单位。而对于未建立任何进程的程序，都不能作为一个独立的单位来运行。

5、为什么进程在进入临界区之前应先执行“进入区”代码？而在退出前又要执行“退出区”代码？请说明

为了实现多个进程对临界资源的互斥访问，必须在临界区之前加一段用于检查临界资源是否正在被访问的代码，如未被访问，该进程可进入临界区对此临界资源进行访问；如正被访问则该进程不能进入临界区访问临界资源。在退出临界区后，执行恢复访问标志的代码为“退出去”，而在退出前执行“退出区”代码主要是为了使其他进程能再访问此临界资源。

6、设 P、Q、R 共享一个缓冲区，P，Q 构成一对生产者和消费者，R 既为生产者又为消费者，使用 P，V 操作实现三个进程同步

```
typedef int semaphore;
semaphore mutex=1,empty=n,full=0;
//设置信号量mutex控制仓库进出，empty表示空仓库的个数，full表示满仓库的个数

void P()
{
    while(true)
    {
        wait(empty);//如果缓冲区已满，则阻塞
        wait(mutex);
        生产一个产品;
        signal(mutex);
        signal(full);//如果消费者被阻塞，则唤醒消费者
    }
}
```

```
void Q()
{
    while(true)
    {
        wait(full);//如果缓冲区为空，则阻塞
        wait(mutex);
        消费者取出一个产品
        signal(mutex);
        signal(empty);//如果生产者已经阻塞，则唤醒生产者
    }
}
```

```
void R()
{
    if(empty==n)//执行生产者的功能
    {
        wait(empty);
        wait(mutex);
        生产一个产品;
        signal(mutex);
        signal(full);
    }
    if(full==n)//执行消费者的功能
    {
        wait(full);
        wait(mutex);
        消费者取出一个产品
        signal(mutex);
        signal(empty);
    }
}
```