一、选择题

1. C
2. B
3. C
4. D
5. B
6. B
7. D
8. C
9. D
10. A
11. B
12. D
13. A
14. C
15. C
16. B
17. 1）0

    2）0

    3）0

    4）0

    5）P(s1);

    6）V(s2); V(s3)

    7）P(s1); P(s2)

    8）V(s4);
18. 给的数可能有问题

(1)（答案不唯一，见注）(2) 不应该 (3) 不存在 (4) 应该 (5) （答案不唯一，见注）

注：(1)P3-P1-(P0,P2,P4 顺序随便)或 P3 -P4-P1-(P0,P2 顺序随便)

(5)P3-P4-P1-(P0,P2 顺序随便)

19. A
20. D
21. B
22. B
23. C
24. A
25. B
26. A
27. D
28. A
29. A
30. D
31. C
32. D
33. D
34. A
35. D
36. C
37. B
38. C
39. B、C 都需要选
40. A

二、简答题

1.答:应该使用用户态线程。
在进程中设计运行时环境，由运行时控制多个线程的调度运行。内核不感知线程的存在，只调度进程。资源分配给进程使用。

2.答:exec 系列函数，如 execlp()等。该系统调用替换进程的正文段，如果成功，没有返回值，如果失败，返回值为-1。

3.答:一般选择运行时间较短的进程，因为这样重新运行的代价较小，另外，程序需要可以多次运行不影响执行结果。还要考虑杀死优先级较低的进程等。

4.解：

（1）由于普通进程的优先级＝优先级基数 + NICE值 + CPU_PENALTY，其中，NICE值和CPU__PENALTY为非负值，所以，只需要将实时进程的优先级设置为小于优先级基数即40）即可。
（2）在普通进程的优先级公式中，NICE值体现静态优先级的概念，CPU_PENALTY体现动态优先级的概念。因为NICE值是由用户指定的，一般不会动态变化的，NICE值越小，进程的优先级越高，这体现了静态优先级的概念。而CPU_PENALTY是随着进程的运行而不断增加，这样使得低优先级的进程，随着时间的推移，终会得到运行，这体现了动态优先级的概念。
（3）由于进程的优先级的取值范围为0-127，而普通进程的优先级中，优先级基数和NICE值相对固定，CPU_PENALTY随着CPU_USAGE增长而增长，如果对CPU_USAGE不加以调整的话，很多进程很快就会达到最低的优先级127，而导致几乎无法运行。所以，CPU_USAGE 每秒钟减半一次。

5.

有三个批处理作业，第一个作业10:00到达，需要执行2小时；第二个作业在10:10到达，需要执行1小时；第三个作业在10:25到达，需要执行25分钟。分别采用先来先服务，短作业优先和最高响应比优先三种调度算法，各自的平均周转时间是多少？
解：
FCFS：执行顺序1->2->3，平均周转时间为(120+170+180)/3=156.7分=2.61小时
SJF：执行顺序1->3->2，平均周转时间为(120+195+120)/3=145分=2.42小时

HRF：执行顺序 1->3->2，平均周转时间为(120+195+120)/3=145 分=2.42 小时

1.有三个批处理作业，第一个作业 10:00 到达，需要执行 2 小时；第二个作业在 10:10 到达，需要执行 1 小时；第三个作业在 10:25 到达，需要执行 25 分钟。分别采用先来先服务，短作业优先和最高响应比优先三种调度算法，各自的平均周转时间是多少？

解：

先来先服务：

（结束时间=上一个作业的结束时间+执行时间

周转时间=结束时间-到达时间=等待时间+执行时间）

按到达先后，执行顺序：1->2->3

作业	到达时间	结束时间	等待时间	执行时间	周转时间	平均周转时间
1	10:00	12:00	0m	120m	120m	156．7m
2	10:10	13:00	110m	60m	170m	
3	10:25	13:25	155m	25m	180m	

短作业优先：

1) 初始只有作业 1，所以先执行作业 1，结束时间是 12:00，此时有作业 2 和 3；
2) 作业 3 需要时间短，所以先执行；
3) 最后执行作业 2

作业	到达时间	结束时间	等待时间	执行时间	周转时间	平均周转时间
1	10:00	12:00	0m	120m	120m	145m
3	10:25	12:25	95m	25m	120m	
2	10:10	13:25	135m	60m	195m	

最高响应比优先：

高响应比优先调度算法既考虑作业的执行时间也考虑作业的等待时间，综合了先来先服务和最短作业优先两种算法的特点。

1) 10:00 只有作业 1 到达，所以先执行作业 1；
2) 12:00 时有作业 2 和 3，
   作业 2：等待时间=12:00-10:10=110m;响应比=1+110/60=2.8；
   作业 3：等待时间=12:00-10:25=95m，响应比=1+95/25=4.8；
   所以先执行作业 3
3) 执行作业 2

作业	到达时间	结束时间	等待时间	执行时间	周转时间	平均周转时间
1	10:00	12:00	0m	120m	120m	145m
3	10:25	12:25	95m	25m	120m	
2	10:10	13:25	135m	60m	195m	

6.

解：用 T 表示周转时间（运行时间+等待时间【开始时间-提交时间】），用 W 表示带权周转时间(周转时间/运行时间)

FCFS的作业调度情况如下：

作业 _	提交时间 _	运行时间 _	开始时间 _	结束时间 _	周转时间 _	带权周转时间 _
1	8.0	1.0	8.0	9.0	1.0	1.0
2	8.5	0.5	9.0	9.5	1.0	2.0
3	9.0	0.2	9.5	9.7	0.7	3.5
4	9.1	0.1	9.7	9.8	0.7	7.0

FCFS的T =（1.0+1.0+0.7+0.7）/ 4 = 0.85W =（1.0+2.0+3.5+7.0）/ 4 =3.375

SJF 的作业调度情况如下：

作业 _	提交时间 _	运行时间 _	开始时间 _	结束时间 _	周转时间 _	带权周转时间 _
1	8.0	1.0	8.0	9.0	1.0	1.0
2	8.5	0.5	9.3	9.8	1.3	2.6
3	9.0	0.2	9.0	9.2	0.2	1.0
4	9.1	0.1	9.2	9.3	0.2	2.0

SJF 的 T=（1.0+1.3+0.2+0.2）/ 4 = 0.675W =（1.0+2.6+1.0+2.0）/ 4 = 1.65

高响应比优先的作业调度情况如下：（响应比=周转时间/运行时间，选择响应比最大的）

作业 _	提交时间 _	运行时间 _	开始时间 _	结束时间 _	周转时间 _	带权周转时间 _
1	8.0	1.0	8.0	9.0	1.0	1.0
2	8.5	0.5	9.0	9.5	1.0	2.0
3	9.0	0.2	9.6	9.8	0.8	4.0
4	9.1	0.1	9.5	9.6	0.5	5.0

高响应比算法的 T=（1.0+1.0+0.8+0.5）/ 4 = 0.825W =（1.0+2.0+4.0+5.0）/ 4 = 3.0

7.解：

（1） _在To时刻存在一个安全序列{P5,P4,P3,P2,P1}，故该状态时安全的。

（2） _在To时刻因进程T2的请求资源（0,3,4）>剩余资源数（2,2,3），所以不能分配。

（3） _此时存在一个安全序列{P4,P5,P3,P2,P1}，即可以将P4申请的资源分配给它。

（4） _此时，可用资源Available已不能满足任何进程的资源请求，因此不能将资源分配给P1。

8.解：

① 系统中资源总量为某时刻系统中可用资源量与各进程已分配资源量之和，所以各种资源总数为(9, 3, 6)。各进程对资源的需求量为各进程对资源的最大需求量与进程已分配资源量之差，

即

$$\begin{bmatrix}3&2&2\\6&1&3\\3&1&4\\4&2&2\end{bmatrix}-\begin{bmatrix}1&0&0\\4&1&1\\2&1&1\\0&0&2\end{bmatrix}=\begin{bmatrix}2&2&2\\2&0&2\\1&0&3\\4&2&0\end{bmatrix}$$

② 若此时P1发出资源请求Request1(1,0,1)，按银行家算法进行检查：
Request1(1,0,1)≤Need1(2,2,2)
Request1(1,0,1)≤Available(2,1,2)
试分配并修改数据结构，资源分配情况如下：

	Allocation			Need			Available
P1	2	0	1	1	2	1	1 1 1
P2	4	1	1	2	0	2	
P3	2	1	1	1	0	3	
P4	0	0	2	4	2	0	

再利用安全性算法检查系统是否安全，可用资源Available(1,1,1)已不能满足任何进程，故系统进入不安全状态，此时系统不能将资源分配给P1。
若此时P2发出资源请求Request2(1,0,1)，按银行家算法进行检查：
Request2(1,0,1)≤Need2(2,0,2)
Request2(1,0,1)≤Available(2,1,2)
试分配并修改数据结构，资源分配情况如下：

	Work			Need			Allocation			Work+ Allocation			Finish
P2	1	1	1	1	0	1	5	1	2	6	2	3	true
P3	6	2	3	1	0	3	2	1	1	8	3	4	true
P4	8	3	4	4	2	0	0	0	2	8	3	6	true
P1	8	3	6	2	2	2	1	0	0	9	3	6	true

从上述分析中可以看出，此时存在一个安全序列{P2,P3,P4,P1}，故该状态是安全的，可以立即将P2所申请的资源分配给它。
③如果②中两个请求立即得到满足后，系统此刻并没有立即进入死锁状态，因为这时所有进程没有提出新的资源申请，全部进程均没有因资源请求没得到满足而进入阻塞状态。只有当进程提出资源申请且全部进程都进入阻塞状态时，系统才处于死锁状态。

9.解：

步骤1、P申请2满足
步骤2、Q申请2满足
步骤3、R申请2满足
步骤4、Q申请2进入阻塞队列
步骤5、R申请2进入阻塞队列
步骤6、P申请2满足

10.答:采用动态调整进程优先级的方法。动态降低长时间占用 CPU 进程的优先级，低优先级的进程的优先级则相对升高，最终得到运行。

11.答:实际常采用的方法: 1、鸵鸟算法。因为处理死锁成本太高，而死锁出现的频率较低，故可以忽略死锁的发生。 2、Spooling 技术。假脱机技术。为临界资源增加一个等待队列，使其好像可以被共享使用，如打印机。 当死锁发生时，杀死运行时间较短的进程，损失较小，因为容易恢复。

12.答:共打印 11 次 hello。