王道操作系统督学营期中考试答案解析

1.

解析：C

分时操作系统强调交互性；实时操作系统强调实时性；批处理操作系统更关注系统的吞吐量、资源利用率；多处理机操作系统强调对多处理机并行工作的有效管理。

2.

解析：D

内中断的产生与当前执行指令有关，外中断的产生与当前执行指令无关，A 正确。

内部异常的检测有 CPU 内部逻辑实现，具体来说，由 CPU 内部的 CU（控制单元）实现，B 正确。

内部异常是 CPU 在执行指令过程中检测到的，C 正确。

只有故障（fault）类的内部异常（如缺页故障），在处理后会返回到发生异常的指令继续执行；陷阱（trap）类的内部异常是由 trap 指令触发的，处理该类异常的过程就是在处理系统调用的过程，系统调用处理完成后，应该执行 trap 指令后序的指令；终止（abort）类的异常通常是不可修复的错误，如整数除以零、非法使用特权指令等，检测到该类异常后，通常 OS 就会终止该进程，不会再返回执行发生异常的指令。D 错误

3.

解析：D

若所有作业都同时到达，则短作业优先调度算法（SJF）具有最小的平均周转时间，所以只需要选择最短作业优先的执行序列即可。

4.

解析：B

信号量表示当前的可用资源数。当信号量 $K \geqslant 0$ 时，表示还有 K 个相关资源可用，且等待该资源的进程数为 0；当信号量 $K<0$ 时，表示有 $|K|$ 个进程在等待该资源，且可用资源数为 0。所以对于该题，可用资源数是 1，等待该资源的进程数是 0。

5.

解析：D

3 个进程，4 个同类资源，每个进程最多同时需要 2 个该类资源。因此，至少能有一个进程可以同时获得 2 个资源并顺序运行下去，系统必然无死锁。

6.

解析：D

当前已分配资源总数 =(3,2,3)+(4,0,3)+(4,0,5)+(2,0,4)+(3,1,4)=(16,3,19)，剩余可用资源数 Available=(18,6,22)-(16,3,19)=(2,3,3)。各进程对资源的需求量 Need 为：

P0=(5,5,10)-(3,2,3)=(2,3,7)

P1=(5,3,6)-(4,0,3)=(1,3,3)

P2=(4,0,11)-(4,0,5)=(0,0,6)

P3=(4,2,5)-(2,0,4)=(2,2,1)

P4=(4,2,4)-(3,1,4)=(1,1,0)

因此初始时只有进程 P1 与 P3 可满足需求，排除 A、C。尝试给 P1 分配资源，则 P1

完成后 Available=(2,3,3)+(4,0,3)=(6,3,6)，无法满足 P0 的需求 (2,3,7)，排除 B。若刚开始给 P3 分配资源，则 P3 完成后 Available=(2,3,3)+(2,0,4)=(4,3,7)，该向量能满足其他所有进程的需求。所以，以 P3 开头的所有序列都是安全序列。

7.

解析：C

若段表、页表存放在内存中，则为了访问内存的某一条指令或数据，将需要访问 3 次内存：

第一次，查找段表获得该段所对应页表的起始地址；

第二次，查找页表获得该页所对应的物理块号，从而形成所需的物理地址；

第三次，根据所得到的物理地址到内存中去访问该地址中的指令或数据

8.

解析：D

采用多级页表，每一级页表的查询都需要访存，因此地址变换速度反而会变慢，A 错误。若采用多级页表，则第一级页表常驻内存，而其他级页表可能尚未调入

内存，因此缺页中断次数反而可能增加，B 错误。页表项中需要记录该逻辑页面对应的物理页框号，而物理页框号的比特数只和物理内存的大小有关，与页表的级数无关，C 错误。若采用一级页表，则为了能够根据页表始址和页号找到对应页表项，整个页表在内存中必须连续存放；若采用多级页表，每一级的页表长度减小，因此页表所占的连续内存空间也可减少，D 正确。

9.

解析：C

存储数组需要 1MB/4KB=256 页，一页可以包含 4KB/8B=512 个页表项，所以 256 页需要一个二级页表就够了。整个过程需访问数组的 256 个页面，每次先查询一级页表，再查询二级页表，因此最多会访问 256*2=512 次页表。

10.

解析：D

在请求分页系统中，只要求将当前一部分页面装入内存，便可以启动作业运行，并不需要一次全部装入，在作业执行的过程中，当访问的页面不存在的时，再通过调页功能将其调入，A 错误。

在请求分页系统中，当要访问的页面不存在的时，便会产生一个缺页中断，OS 会将该页调入内存中，当内存中有空闲页框时候，将需要的页面直接调入空闲页框中。当内存中没有空闲页框时，才需淘汰掉一个页面，然后将需要调入的页面调入，B 错误。

淘汰一个页面时，如果该页面没有被修改的话，便不用写回外存，C 错误。

页表构成：页号(隐含)+页框号+状态位 p+访问字段 A+修改位 M+外存地址 L

状态位 P：标记该页是否已被调入内存中，用于判断是否触发缺页异常

访问字段位 A：记录本页在一段时间内被访问的次数，或页面被调入的时间等，供页面置换算法参考

修改位 M：标记该页面在调入内存后是否被修改，当页面被淘汰时，若页面数据没有修改，则不用写回外存

外存地址 L：该页在外存的地址，供写回外存和从外存中调入该页时参考

应用题 1

答案：

```
semaphore mutex=1,white=0,black=0;
                //mutex 保证取子计数时互斥，white 和 black 表示是否已经取完白子和黑子
int count=0,White=Black=0;                    //int 型的 White 和 Black 表示白子和黑子数
process A()
{
    while (true)
    {
        P(mutex);
        if (Getblack()) {count++;Black++;}
            else {V(black);V(mutex);break;}        //这里一定要在 exit 退出之前 V(mutex)
        V(mutex);
    }
}
process B()
{
    while (true)
    {
        P(mutex);
        If (Getwhite()) {count++;white++;}
            else {V(white);V(mutex);break;}        //和上面一样，保证退出循环之前
V(mutex)
        V(mutex);
    }
}
Process C()
{
    P(black);
    P(white);
    Printf();
}
```

答案：

（1）

FIFS：　共缺页 15 次，缺页率为 15/20=3/4

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
○	○	○	○		○	○	○	○	○	○			○	○			○	○	○

LRU：　共缺页 12 次，缺页率为 12/20=3/5

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	0	0	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	1	1	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
○	○	○	○		○		○	○	○	○			○		○		○		

Clock：　共缺页 15 次，缺页率为 14/20=7/10

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	4	3	3	3	3	0	0	1	1	1
	0	0	0	0	0	0	0	2	2	2	2	2	1	1	1	1	7	7	7
		1	1	1	3	3	3	3	3	0	0	0	0	2	2	2	2	0	0
○	○	○	○		○		○	○		○	○		○	○	○		○	○	

Clock，完成时的使用位（颜色表指针）

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0	1
	1	1	0	1	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1
		1	0	0	1	1	0	0	1	1	1	1	0	1	1	1	0	1	1

（2）

FIFS：　共缺页 15 次，缺页率为 11/20

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	7	7	7
		1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
○	○	○	○	○	○		○			○			○	○			○		

LRU： 共缺页 8 次，缺页率为 8/20=2/5

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
○	○	○	○		○		○						○				○		

Clock： 共缺页 8 次，缺页率为 9/20

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	7	7	7
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
○	○	○	○		○		○						○	○			○		

Clock，完成时的使用位（颜色表指针）

7	0	1	2	0	3	0	4	2	3	0		3	2	1	2	0	1	7	0	1
1	1	1	1	1	1	1	1	1	1	1		1	1	0	1	1	1	1	1	1
	1	1	1	1	0	1	0	0	0	1		1	1	0	0	1	1	0	1	1
		1	1	1	0	0	1	1	1	1		1	1	0	0	0	0	1	1	1
			1	1	0	0	0	1	1	1		1	1	1	1	1	1	1	1	1

（3）LRU 和 Clock 算法缺页率随物理块数增加而减少，因为他们都是采用堆栈类算法没有 Belady 异常（FIFS 不行）