Sample Exam Questions

一、简答题

1、（5 分）在支持分页的操作系统中，页面大小通常设置为 1KB～4KB。为什么不使用更小或更大的页面大小？请分别给出一个可能的原因。

参考答案：页面太小会导致进程所需的页面数增加，页表会变得很大，在多级页表下需要多次的内存访问来进行地址转换。页面太大会导致内部碎片的增加，浪费内存空间。

2、（5 分）非安全状态一定会导致死锁吗？为什么？

参考答案：不一定。非安全状态下，只要进程不按照最大需求申请资源，那么可以一直保持非死锁状态。此外，在运行过程中，进程甚至可能释放某些资源，从而解除非安全状态。

3、（5 分）Read-Copy-Update（RCU）的宽限期（Grace Period）的作用是什么？

参考答案：在 RCU 更新数据过程中，如果新数据发布后立即删除旧数据，此时还在访问旧数据的进程就会丢失信息（既无法看到完整的旧数据，也无法看到新数据），使用宽限期可以指明何时可以删除旧数据。

4、（5 分）在 Unix 文件系统的设计中，系统调用 open 是必须的吗？如果没有该系统调用会对文件系统的使用产生什么影响？

参考答案：如果没有 open 操作，那么每次调用 read 和 write 等操作时都需要指明目标文件名，然后操作系统需要在磁盘寻找文件对应的 inode。另一方面，同一个文件可能以不同的方式打开，直接使用文件名调用 read 和 write 无法对此进行有效区分。

二、应用题

1、（6 分）考虑如下使用 Pthreads API 的代码，该程序在 LINE C 和 LINE P 处的输出分别是什么？

```
int value = 0;
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
      pid_t pid;
      pthread_t tid;
      pthread_attr_t attr;

      pid = fork();

      if (pid == 0)    /* child process */
      {
            pthread_attr_init(&attr);
            pthread_create(&tid, &attr, runner, NULL);
            pthread_join(tid, NULL);
            printf("CHILD: value = %d\n", value);    /* LINE C */
      }
      else if (pid > 0)    /* parent process */
      {
             wait(NULL);
             printf("PARENT: value = %d\n", value);    /* LINE P */
      }
}

void* runner(void* param)
{
      value = 5;
      pthread_exit(0);
}
```

参考答案：LINE C 处的输出为 5，LINE P 处的输出为 0。

2、（6 分）在时间片轮转（RR）策略的设计实现中，我们通常会维护一个运行队列，该队列的元素是对任务的引用（指针），每次调度任务会运行固定的时间片长度。试想，如果在 RR 策略的队列中加入对同一个任务的多个引用，那么这样的设计会带来什么影响？尝试通过修改基本的 RR 算法达到同样的效果但避免使用多余的指针。

参考答案：这个设计会使得增加引用的任务提升了优先级，因为其可以获得更多的时间片。

修改方法：可以尝试对不同的任务赋予不同的时间片数量。

3、（8 分）某系统使用银行家算法来避免死锁。假设该系统中有 T1、T2、T3、T4 四个线程需要获取共享资源 A 与 B，某一时刻系统的状态如下所示：

线程	当前分配		最大需求		可用资源	
	A	B	A	B	A	B
T_1	0	3	4	6	2	5
T_2	2	0	4	1		
T_3	1	0	4	8		
T_4	7	1	12	5		

1) 当前系统是否处于安全状态？如果处于安全状态，请给出一个安全序列。

2) T1 向系统申请 2 个 A 资源和 3 个 B 资源，系统是否可以满足该请求？

3) T3 向系统申请 2 个 B 资源，系统是否可以满足该请求？

参考答案：

1) 处于安全状态，存在安全序列 T2-T1-T3-T4

2) 如果满足该请求，系统进入不安全状态，因此不能立刻满足

3) 如果满足，系统仍然是安全状态，可以立即满足

4、（8 分）假设某磁盘有编号为 0~39 的 40 个磁道（Track），当磁头位于第 11 号磁道时顺序来到如下请求（磁道号）：1、36、16、34、9、12、13。请给出 FCFS（First Come First Service）、SSTF（Shortest Seek Time First）和电梯算法（SCAN）这三种磁盘驱动调度算法的访问磁道顺序，并计算出它们各自要来回穿越多少磁道。

参考答案：

FCFS：11-1-36-16-34-9-12-13，Sum = 112 (10+35+20+18+25+3+1)

SSTF：11-12-13-16-9-1-34-36，Sum = 55 (1+1+3+7+8+33+2)

电梯调度算法（由小到大）：11-12-13-16-34-36-9-1，Sum = 60 (1+1+3+18+2+27+8)

电梯调度算法（由大到小）：11-9-1-12-13-16-34-36，Sum = 45 (2+8+11+1+3+18+2)

5、（10 分）假设有如下一个 Unix 文件系统：

- 磁盘块大小 1 KB，每个磁盘分区分配 512 个块用于存储空闲空间管理信息（bitmap），分配 4096 个块用于存储文件 inode 信息（array of inodes）；
- 每个 inode 存储用户 ID（2 B）、三个时间戳（每个 4 B）、类型和保护位（4 B）、引用计数（2 B）、文件大小（4 B）、以及文件存储的索引结构信息；
- 文件 inode 的索引结构使用 8 个直接指向磁盘数据块的索引项（指针）、1 个指向一级间接索引块的索引项、以及 1 个指向二级间接索引块的索引项，每个索引项 4 B；
- 每个目录项存储文件名及其对应的 inode 号，其中文件名以 14 B 固定长度存储；

根据上述文件系统设计，回答下列问题：

1） 该文件系统可支持的最大文件大小为多少？

2） 该文件系统的单个目录最多可包含多少个文件？

3） 为了在存储 array of inodes 的磁盘块出现损坏时仍能恢复其中的信息，一种方法是在磁盘分区中额外存储 array of inodes 的一个备份。这样一种设计对上述文件系统的性能会有何影响？

参考答案：

1） 每个间接索引块包含 1024/4 = 2^8 个索引项，最大支持文件大小为 8 * 1KB + 2^8 * 1KB + 2^8 * 2^8 * 1KB = 2^13 + 2^18 + 2^26 B = 8KB + 256KB + 64MB

2） 每个 inode 结构需要 4 + 12 + 2 + 2 + 4 + 10*4 = 64 B 存储空间，根据文件系统布局，总共可有 (2^12 * 2^10) / 2^6 = 2^16 个 inode，对应需要 2 B 存储 inode 号；
根据目录结构，每个目录项需要 14 + 2 = 16 B，目录以文件方式存储，因此最多 (2^13 + 2^18 + 2^26) / 2^4 = 2^9 + 2^14 + 2^22 个文件。该数值小于 2^16，因此单个目录最多可包含 2^16 个文件

3） 对 inode 的写操作不可避免地需要同时对 inode 备份进行更新，这会使得新建文件、删除文件、增加文件长度等操作的性能下降。与之相比，对 inode 的读操作仅需从其中一份进行读取，此时如果有专门为此设计的磁盘调度算法的话（移动到最近的 inode 块），能提高对 inode 读操作的性能。