操作系统 期中考试

类型：独立完成开卷考试，不得参考他人（包括往届）答案
截止日期：本学期考试周前最后一周周日（含）
上交方式：打印，手写，扫描为 pdf 后交给助教，文件格式“期中-姓名-学号.pdf”

姓名： 陈鹏雯　　**学号：** 202200130048

诚信声明：我承诺本开卷考试由本人独立手写扫描完成，没有参考他人的答案。（在下面抄写）

我承诺本开卷考试由本人独立手写扫描完成，没有参考他人的答案。

一、单项选择题（10x2 = 20）

1. 操作系统的开发最需要考虑（ A ）B.
   A.应用场景　　B.用户体验
   C.生态与兼容性　　D.以上全错

原则：OS 最高设计目标，由所在领域的需求决定。

2. 操作系统的各个组件的功能的低层次代码文本又叫做其（ B ）。
   A.设计：高层次抽象，逻辑功能　　B.实现
   C.策略：对资源进行管理的方法和政策　　D.机制：为实现策略使用的工具和手段。

3. 一般而言，多线程进程中的线程不能共享的是（ D ）。
   A.代码段　　B.数据段
   C.堆段　　D.栈段

TCB 多在内核.

4. 下列有关基于时间片轮转法的线程调度的叙述中，错误的是（ B ）。
   A.时间片越短，线程切换的次数越多，系统开销也越大 ✓
   B.当前线程的时间片用完后，该线程状态由执行态变为阻塞态 ✗ 就绪.
   C.时钟中断发生后，系统会修改当前线程在时间片内的剩余时间 ✓
   D.影响时间片大小的主要因素包括响应时间、系统开销和进程数量等

5. 下列选项中，无论采取何内核结构，最不可能在用户模式执行的是（ D ）。
   A.命令解释程序　　B.缺页处理程序
   C.进程调度程序　　D.时钟中断处理程序

硬件资源分配. 内存管理.

6. 用户程序发出磁盘 I/O 请求后，系统的正确处理流程是（ B ）。
   A.用户程序→系统调用处理程序→中断处理程序→设备驱动程序
   B.用户程序→系统调用处理程序→设备驱动程序→中断处理程序

C.用户程序→设备驱动程序→系统调用处理程序→中断处理程序
D.用户程序→设备驱动程序→中断处理程序→系统调用处理程序

7. 操作系统课程中介绍的，负责在系统上电时提供万年历时间的是（ ）
A.时钟源 B.定时器
C.晶体振荡器 D.实时时钟

8. 在下列选项中，不能显著改善磁盘 I/O 性能的是（ B ）。
A.重排 I/O 请求次序 ✓
B.在一个磁盘上设置多个分区
C.预读取和写合并 ✓
D.优化文件物理块的分布 ✓

9. 在对同一个数据结构的同一处的操作中，若颠倒次序则很可能会破坏数据结构的是（ ）。
A.读-读
B.读-写
C.写-读
D.写-写

10. 在下列 IPC 方法中，不能用于任意两个进程之间通信的是（ ）。
A.无名管道
B.信号量 ✓
C.消息队列 ✓
D.信号

二、填空题（10x3 = 30）

1. 在程序的两种基本链接方法中，性能比较高的是 动态链接

2. 微内核（类型的）操作系统中，应用程序必须请求守护进程中的策略分配敏感资源，守护进程则互相通信或转而使用内核提供的机制完成这些分配操作。

3. 一般地，线程优先级数字越 小 ，线程优先级越高，但也存在例外。

4. 线程的三个基本状态是 就绪态、运行态、阻塞态（每项 1 分）。

5. 能对敏感资源进行操作、仅能在内核模式下执行的指令叫 特权指令。

6. 若给进程分配的页框数量远少于其当前工作集，则缺页率会 增加 （增长/下降；1 分），这种现象叫做 抖动 （2 分）。

7. 在课上介绍的几种机械磁盘寻道算法中，会造成饥饿的是 最短寻道时间优先

8. 磁带等介质使用的文件系统的文件分配方法是 连续分配

9. 文件的索引节点(inode)中一般包含 文件类型、文件大小、文件权限（列出任意三项，每项得 1 分；错列则本题不得分）。

10. 如果系统进入不安全状态，而且其中的指令流（自私）在得到自己可能请求的全部资源之前绝不释放占有的资源，只当到结束后才一齐释放，则系统必定死锁。

三、名词解释（5x4 = 20）

1. 操作系统（4 分；4 个要点，每个各 1 分）

操作系统是管理计算机硬件和软件资源的应用程序，操作系统能够对系统资源进行调度、进行存储、设备管理，还对计算机的保护和安全进行负责，提供给用户和其他软件方便的接口和环境是最基本的系统软件。

2. 文件系统的本质特征（2 分）与索引节点表（2 分）

文件系统：将文件中的文件块按照一定的方法最终映射到物理设备上的物理块，并在物理存储器特性的限制下提供尽可能高的信息管理效率。偏重信息存储、更底层、更原始。提供的功能更少。

索引节点表：一个线性表存放索引节点，通过索引节点知道文件除文件名以外的其他信息。

3. 平均等待时间（2 分）与响应比（2 分）的定义式

$$\text{平均等待时间} = \frac{\text{多线程从就绪系到运行态等待时间之和}}{\text{线程数}} = \frac{\sum_{i=1}^{n} W_i}{N}.$$

$$\text{响应比} = \frac{\text{周转时间}}{\text{运行时间}} = \frac{T}{R} = 1 + \frac{W}{R}.$$

4. 同步接口（2）与阻塞式接口（2 分）的定义

同步接口：用同一个接口操作来发起 I/O 请求和接收 I/O 结果，当接口返回时 I/O 结果必已知。

阻塞接口：当设备无法即时完成操作或返回消息时，在接口上请求 I/O 操作的线程将阻塞，直到返回数据。一定需要操作系统介入，阻塞接口一定是同步接口。

5. 死锁的四个条件（各 1 分）

① 互斥条件：存在有一定互斥性的资源。

② 保持请求：不主动放弃已经持有的资源。

③ 无法剥夺：不允许相互剥夺资源。

④ 循环等待：循环等待资源释放，不能进展。

四、问答题（2x10 = 20）

1. 一个网络服务器要同时响应多个客户的多个连接请求，既可以采用每个请求启动一个单独的进程也可以采用单进程中每个请求启动一个线程。问：
（1）采用多进程方法有什么优势（2 分）和劣势（2 分）？
（2）采用单进程多线程方法有什么优势（1 分）和劣势（1 分）？
（3）你能想出一种折中方法吗（2 分）？
（4）如果希望更高效地利用服务器的资源，降低线程切换开销，在（2）的基础上还可以如何编写程序（2 分）？

(1) 优：方便对每个用户单独操作，有独立的堆栈空间．每个进程之间互不干扰．

缺：增加了操作系统负担，切换进程需要的开销大，同时为每个进程分配空间不仅需要单独管理，占用的资源也增加了．

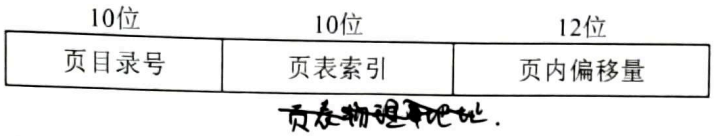
(2) 优：每个线程共享进程资源，切换进程时开销较小．

劣：共享资源可能会引起线程间的资源竞争，容易引发死锁．

(3) 可以将多个线程分给多个进程管理，每个进程管理多个线程．

(4) ~~单进程中又可以分为用户级线程和内核级线程，每个内核级线程可以对应多个用户级线程也就是多个~~ 进程中的指令流与线程进行多对多的对应，每个线程分配多个指令流．任何一个不阻塞的线程都可以运行任何一个不阻塞的指令流，同一线程中的多个指令流共享一份执行时间，共享线程资源，被当作一个对象处理．

2. 某 32 位 x86 系统按字节编址，采用二级页表的分页存储管理方式，虚拟地址格式如下所示。问：



（1）页（1 分）和页框（1 分）的大小各为多少字节？进程的虚拟地址空间大小是多少，它可以分成多少页（2 分）？

（2）假定页目录项和页表项均占 4 个字节，对于一个完全映射了所有虚拟内存空间的进程，其页目录占多少页（2 分）？页表共占多少页（2 分）？

（3）若连续三条访存指令分别访问的虚拟地址为 0100 0000H、0102 0000H、01000231H，则进行地址转换时共访问多少个不同的二级页表（1 分）？共访问多少次二级页表（1 分）？

(1) $2^{12} = 4096$ B 页和页框都为 4096 字节.

2^{10} 个二级页表，每个二级页表有 2^{10} 个页，每个页大小 2^{12}.

∴ 虚拟地址空间：2^{32} B.

分为 2^{20} 页.

(2) 每页 2^{12}，每项 4 个字节，一页有 2^{10} 项.

页目录共 2^{10} 项 $2^{10} < 2^{12}$，占一页 页目录 $2^{10} / 2^{10} = 1$ 页

页表共 2^{20} 项 $2^{20} / 2^{12} = 2^8 = 256$ 页 页表 $2^{20} / 2^{10} = 2^{10} = 1024$ 页

(3) 0000 0001 00, 00 0000 0000, ...

前三位都是 010H，即都访问同一个二级页表 0000 0001 00B，

也就是 二级页号看中间 10 位，分别是 000H，020H，000H.

即 2 个不同的二级页表，共访问三次二级页表.

五、程序编写（10）

有一间 100 个座位的空教室供 1 班和 2 班两个班的同学上自习使用，教室只有一个出入口，每次只允许一个同学通过。若无空座，则欲进入教室的同学在门口等待，直到有同学离开方可进入。而且，为公平起见，规定教室中 1 班的同学数与 2 班的同学数相差不超 10 个。

（1）请利用①互斥锁机制及其 lock 和 unlock 操作与条件变量机制及其 wait、wakeup 和 wakeup_all 操作或②信号量及其 acquire 和 release 操作（任选①②其一）并采用类 C 语言编写伪代码解决上述问题。假设 1 班同学调用的函数名为 first_enter 和 first_exit，2 班同学调用的函数名为 second_enter 和 second_exit，其它变量名称自选。（不考察语法正确性；6 分）

（2）额外要求-管程封装：将（1）问的解决方案封装成管程类 class classroom，并给出其类定义、构造函数和析构函数。所有同步和互斥机制都要求在构造函数中初始化（init），并在析构函数中删除（delete）。（1 分）

（3）额外要求-参数化：在（2）问的解决方案中，要求在创建管程时可以选择任意的教室大小 max 和任意的同学相差数 diff（1 分）。

（4）额外要求-条件变量：在（1）问的解决方案中，选择条件变量机制（1 分）。

（5）额外要求-避免惊群：在（3）问的解决方案中，不产生惊群效应（1 分）。慎用全唤醒.

(1) 选①，完成(4).

满 → 1.2出

1空 → 1进 ； 2空 → 2进.

1多 → 1出.2进 ； 2多 → 2出.1进.

```
mutex_t door ; 互斥锁.
int num_1 = 0 , num_2 = 0; // 1.2班人数
condition_t ~~class~~ full = empty = more_1 = more_2 = cond_init ;
        ~~= full_2 = empty~~ empty_1 = empty_2
```

```
first_enter(){
    while(1){
        lock(&door);
        while ( num_1 + num_2 ≥ 100 )
                    ~~num_1 - num_2 ≥ 10~~
            cond_wait ( &full , &door);
        while ( num_1 - num_2 > 10 )
            cond_wait ( &more_1 , &door);
        num_1 ++ ;
        cond_signal ( &more_2 , &door);
        cond_signal ( &empty , &door);
        unlock (&door);
    }
}
```

管程封装

```
first_leave(){
    while(1){
        lock(&door);
        while ( num_1 <= 0 )
            ~~wait~~ cond_wait ( &empty_1 , &door);
        num_1 -- ;
        cond_signal ( &full , &door);
        cond_signal ( &more_1 , &door);
        unlock (&door);
    }
}
```

```
second_enter(){
    while(1){
        lock(&door);
        while(num_1 + num_2 >= (w))
            cond_wait(&full, &door);
        while(num_2 - num_1 > (0))
            cond_wait(&more_2, &door);
        num_2++;
        cond_signal(&more_1, &door);
        cond_signal(&empty_2, &door);
        unlock(&door);
    }
}
```

```
second_leave(){
    while(1){
        lock(&door);
        while(num_2 <= 0)
            cond_wait(&empty_2,
                      &door);
        num_2--;
        cond_signal(&more_2, &door);
        cond_signal(&full, &door);
        unlock(&door);
    }
}
```

封装后自动加锁和释放锁.

(3).(4) 封装管程.参数(x.

```
class classroom{
    condition_t full, empty_1, empty_2, more_1, more_2;
    int num_1=x, num_2=x;
    int max, diff seat, d;
    mutex_t door;
    void init(int max, int diff){
        seat = max;
        d = diff;
        num_1 = 0, num_2 = 0;
        mutex_init(&door, NULL);
        cond_init(&full, NULL);
        cond_init(&empty_1, NULL);
        cond_init(&empty_2, NULL);
        cond_init(&more_1, NULL);
        cond_init(&more_2, NULL);
    }
```

```
    void delete(){
        mutex_destroy(&door);
        cond_destroy(&full);
        cond_destroy(&empty_1);
        cond_destroy(&empty_2);
        cond_destroy(&more_1);
        cond_destroy(&more_2);
    }
};

void first_enter();
void first_leave();
void second_enter();
void second_leave();
// 把lock和unlock之间的代码封装封装.
```