操作系统课考试

姓名_____ 证件类型_____ 证件号_____ 考试日期_____

成绩_____

1. 考试时长为 120 分钟；共有 11 个题目，总分为 100 分。其中，第一题（诚信承诺书）有 1 小题(2 分)，第二题（临界区）有 2 小题(4 分)，第三题（进程管理 ）有 4 小题(10 分)，第四题（多处理机调度）有 2 小题(6 分)，第五题（嵌套异常）有 1 小题(8 分)，第六题（中断+调度+进程管理）有 1 小题(6 分)，第七题（页面替换算法）有 4 小题(16 分)，第八题（锁的实现）有 3 小题(10 分)，第九题（基于信号量的同步互斥）有 1 小题(14 分)，第十题（内存映射）有 5 小题(14 分)，第十一题（文件系统）有 5 小题(10 分)。请注意合理分配时间。

2. 考试采用“禁止翻题”方式，只有完成前一个题目的作答后，才能看到下一个题目，并且不能修改已完成的作答。请注意提交结果前的检查，以减少笔误。建议按题目分值占比来大致分配考试时间，并留一定的时间余量。

3. 考试中只允许在易考客户端上作答，作答时不能切换到其他应用。

4. 手机只用于监考，考试中禁止使用手机的监考以外的任何功能。

5. 如果对题目表达有歧义或认为题目中的条件不足够，请明确描述你的理解或增加你认为需要的条件，据此进行作答。

6. 问题中的“回答字数限制”是建议，不是强制要求。

一、诚信承诺书 （共 1 题 共 2 分）

1.（2 分）我郑重承诺：

1. 在考试期间，不使用、提供或接受未经授权的任何帮助或信息，不请人代考或者代替别人考试，按要求独立答卷，不与他人进行交流。

2. 严格遵守校规校纪，诚信考试！若有违反考试纪律行为，同意按照清华大学关于考试的管理规定处理。

3. 愿意配合监考教师随时通过网络或者其他方式进行抽查。

4. 保护清华大学知识产权，绝不保存、散播本次考试题目。

请在作答区输入“我承诺，严格遵守校规校纪，诚信考试！”，并写上自己的名字和学号。

二、临界区 （共 2 题 共 4 分）

2.（2 分）为了形成临界区，在单核机器上可以使用关中断，在多核机器上可以使用自旋锁。请回答下列问题。

在单核机器中使用自旋锁可以吗？为什么？（回答字数 150 字）

3.（2 分）为了形成临界区，在单核机器上可以使用关中断，在多核机器上可以使用自旋锁。请回答下列问题。

在多核机器上使用关中断可以吗？为什么？（回答字数 150 字）

三、进程管理 （共 4 题 共 10 分）

4.（3 分）什么是孤儿进程？（回答字数 100 字）

5.（3 分）什么是僵尸进程？（回答字数 100 字）

6.（2 分）请描述在何种情况下有进程会成为孤儿进程。（回答字数 100 字）

7.（2 分）请描述在何种情况下有进程会成为僵尸进程。（回答字数 100 字）

四、多处理机调度 （共 2 题 共 6 分）

8.（3 分）

在多处理机调度中，就绪任务队列的维护有两种策略：

单队列：使用无锁数据结构，维护一个全局队列

多队列：每个核有一个自己的局部队列

这两种方案相比，分别有什么好处和不足？（回答字数 150 字）

9.（3 分）在多处理机调度中，就绪任务队列的维护有两种策略：

单队列：使用无锁数据结构，维护一个全局队列

多队列：每个核有一个自己的局部队列

提出一种新的改进方案，避免这两种方案各自的不足之处。（回答字数 150 字）

五、嵌套异常 （共 1 题 共 8 分）

10.（8 分） 由于内核实现代码的错误，导致内核在执行缺页（page fault）处理例程（响应用户进程导致的第一次缺页异常）的过程中再次（第二次）发生缺页异常。假定该内核其他部分编程正确且支持内核态中断。请描述第二次缺页异常产生之后的内核执行过程，以及从第一次缺页异常产生后开始的整个过程中内核栈变化和寄存器的状态变化的情况。（回答字数 200 字）

六、中断+调度+进程管理 （共 1 题 共 6 分）

11.（6 分）假定在一台装有类 UNIX 通用操作系统的单核计算机上，任意一个中断从产生到响应处理结束，耗时 1 ms；OS 调度算法是时间片轮转调度算法，调度时间片为 100 ms ；时钟中断周期 10 ms；系统中有 3 个处于就绪态的进程。一个程序从十点整开始运行，恰好 60,000 ms 后运行结束，在运行过程中产生了 6,100 次中断。

请根据以上信息回答：该程序的实际占用 CPU 的真实运行时间是否可以计算得出？若可以，给出计算过程和具体结果（单位 ms ，保留整数）；若不可以，给出理由。（回答字数 150 字）

七、页面替换算法 （共 4 题 共 16 分）

12.（4 分）

关于局部页面替换算法：最优算法（OPT）/先进先出算法（FIFO）/最近最久未使用算法（LRU）/时钟（clock）页面置换算法。

请问，什么是 Belady 现象？在上述四种算法中，哪些算法可能存在 Belady 现象？（回答字数 100 字）

13.（4 分）

关于局部页面替换算法：最优算法（OPT）/先进先出算法（FIFO）/最近最久未使用算法（LRU）/时钟（clock）页面置换算法。

请问，设一个页面访问序列为 0,1,4,2,3,4,1,0,3,2，且物理页帧中最多能容纳 4 个页面，初始时物理页帧为空，请用 FIFO 算法进行模拟，给出每一次访问的情况以及访问后物理页帧的状态，最终给出总缺页次数。

14.（4 分）

关于局部页面替换算法：最优算法（OPT）/先进先出算法（FIFO）/最近最久未使用算法（LRU）/时钟（clock）页面置换算法。

请问，设一个页面访问序列为 0,1,4,2,3,4,1,0,3,2，且物理页帧中最多能容纳 4 个页面，初始时物理页帧为空，请用 LRU 算法进行模拟，给出每一次访问的情况以及访问后物理页帧的状态，最终给出总缺页次数。

15.（4 分）

关于局部页面替换算法：最优算法（OPT）/先进先出算法（FIFO）/最近最久未使用算法（LRU）/时钟（clock）页面置换算法。

请问，从应用程序和操作系统在通用 CPU 上实际运行的角度看，如果操作系统采用 LRU 页面置换算法，那相比于采用 FIFO 页面置换算法，其在整体系统执行效率上的结果上一定更好吗？说明理由。（回答字数 150 字）

八、锁的实现 （共 3 题 共 10 分）

16.（3 分）

在实现各种锁时，需要让一些 Read-Modify-Write 操作同时进行，且不能被其他线程打断，这就需要用硬件提供的原子操作指令来实现。常见的原子操作指令有

TestAndSet/CompareAndSwap/FetchAndAdd 等，它们的功能描述如下(类 C 伪码实现，每个函数体都是原子的，不能被打断)：

```
bool TestAndSet(bool *ptr) {
    if (!*ptr) {
        *ptr = true;
        return false;
    } else {
        return true;
    }
}

bool CompareAndSwap(int *ptr, int oldval, int newval) {
    if (*ptr == oldval) {
        *ptr = newval;
        return true;
    } else {
        return false;
    }
}

int FetchAndAdd(int *ptr, int value) {
    int newval = *ptr + value;
    *ptr = newval;
    return newval;
}
```

用 TestAndSet 原子指令，实现自旋锁。请用类 C 伪码形式补全下面代码中的 `lock` 结构和获取锁`lock_acquire()`、释放锁`lock_release()` 函数。

```
struct lock {
    // Your code 1
};

struct lock spin;

void lock_acquire(struct lock* spin) {
    // Your code 2
}

void lock_release(struct lock* spin) {
    // Your code 3
}
```

17.（3 分）

在实现各种锁时，需要让一些 Read-Modify-Write 操作同时进行，且不能被其他线程打断，这就需要用硬件提供的原子操作指令来实现。常见的原子操作指令有 TestAndSet/CompareAndSwap/FetchAndAdd 等，它们的功能描述如下(类 C 伪码实现，每个函数体都是原子的，不能被打断)：

```
bool TestAndSet(bool *ptr) {
    if (!*ptr) {
        *ptr = true;
        return false;
    } else {
        return true;
    }
}

bool CompareAndSwap(int *ptr, int oldval, int newval) {
    if (*ptr == oldval) {
        *ptr = newval;
        return true;
    } else {
        return false;
    }
}

int FetchAndAdd(int *ptr, int value) {
    int newval = *ptr + value;
    *ptr = newval;
    return newval;
}
```

用 CompareAndSwap 原子指令，实现自旋锁。请用类 C 伪码形式补全下面代码中的 `lock` 结构和获取锁`lock_acquire()`、释放锁`lock_release()` 函数。

```
struct lock {
    // Your code 4
};

struct lock spin;

void lock_acquire(struct lock* spin) {
    // Your code 5
}

void lock_release(struct lock* spin) {
    // Your code 6
}
```

18.（4 分）

在实现各种锁时，需要让一些 Read-Modify-Write 操作同时进行，且不能被其他线程打断，这就需要用硬件提供的原子操作指令来实现。常见的原子操作指令有 TestAndSet/CompareAndSwap/FetchAndAdd 等，它们的功能描述如下(类 C 伪码实现，每个函数体都是原子的，不能被打断)：

```
bool TestAndSet(bool *ptr) {
    if (!*ptr) {
        *ptr = true;
        return false;
    } else {
        return true;
    }
}

bool CompareAndSwap(int *ptr, int oldval, int newval) {
    if (*ptr == oldval) {
        *ptr = newval;
        return true;
    } else {
        return false;
    }
}

int FetchAndAdd(int *ptr, int value) {
    int newval = *ptr + value;
    *ptr = newval;
    return newval;
}
```

使用 `FetchAndAdd` 原子指令原语可实现一种 `ticket lock`，类 C 伪码形式的代码如下。试分析这种锁与自旋锁相比有何优势？ （回答字数 150 字）

```
struct lock {
    int ticket;
    int turn;
};

struct lock tlock;

void lock_acquire(struct lock* tlock) {
    int myturn = FetchAndAdd(tlock->ticket, 1);
    while (tlock->turn != myturn);
}

void lock_release(struct lock* tlock) {
    tlock->turn += 1;
}
```

九、基于信号量的同步互斥 （共 1 题 共 14 分）

19.（14 分）假设某仓库可存放 A、B 两种物品，其容量无限大；任何时刻只允许一个物品进行仓库进行入库操作；要求仓库中 A 物品数量 X 和 B 物品数量 Y 满足不等式： -M ≤（X - Y）≤ N，其中 M 和 N 为正整数。请用信号量机制实现 A、B 两种物品入库的处理过程，要求能做到正确且高效的同步与互斥。

请说明所定义的信号量的含义和初始值，描述需要进行互斥处理的各种行为，描述需要进行同步处理的各种行为；要求用类 C 语言伪代码实现，并给出必要的简明代码注释。

十、内存映射 （共 5 题 共 14 分）

20.（6 分）

假定在某基于二级页表的虚拟页式存储，具有 4GB 物理内存的 32 位计算机系统下，32 位虚拟地址从高到低被划分如下， 10 位页目录序号、10 位页表序号和 12 位页内偏移。页目录和页表大小均为 4096 字节，页目录项和页表项大小为 4 字节。使用自映射方式组织页表。本题表述中，用 `flags` 表示页表/页目录项的各种标志位，占低 12 位，且 CPU 不基于 `flags`的信息来判断某项是页表项还是页目录项。要求回答中所有地址均使用用十六进制表示，如 `0x1edc8000`。

除了自映射，举出一个其他页表组织方式。与自映射相对比，你给出的其他页表组织方式有什么优缺点？要求至少一个优点和一个不足。（回答字数 150 字）

21-22.（共 4 分）

假定在某基于二级页表的虚拟页式存储，具有 4GB 物理内存的 32 位计算机系统下，32 位虚拟地址从高到低被划分如下， 10 位页目录序号、10 位页表序号和 12 位页内偏移。页目录和页表大小均为 4096 字节，页目录项和页表项大小为 4 字节。使用自映射方式组织页表。本题表述中，用 `flags` 表示页表/页目录项的各种标志位，占低 12 位，且 CPU 不基于 `flags`的信息来判断某项是页表项还是页目录项。要求回答中所有地址均使用用十六进制表示，如 `0x1edc8000`。

假定页目录起始物理地址是 0x1edc8000，并且物理地址 0x1edc8ff0 (=0x1edc8000+1020*4) 开始的页目录项是自映射项。则此处自映射项的内容是 `_____ ___ | flags`，该自映射项的虚拟地址是`_____ ____`？

23-24.（共 4 分）

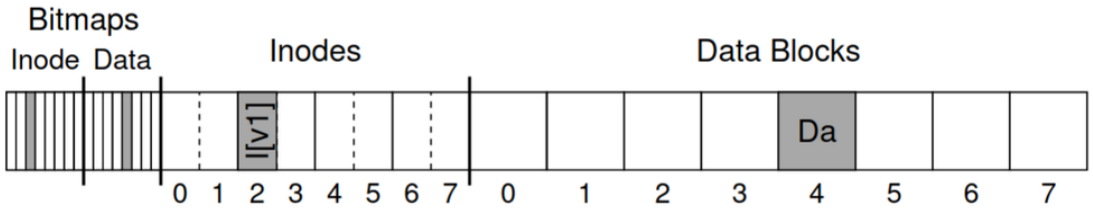
假定在某基于二级页表的虚拟页式存储，具有 4GB 物理内存的 32 位计算机系统下，32 位虚拟地址从高到低被划分如下， 10 位页目录序号、10 位页表序号和 12 位页内偏移。页目录和页表大小均为 4096 字节，页目录项和页表项大小为 4 字节。使用自映射方式组织页表。本题表述中，用 `flags` 表示页表/页目录项的各种标志位，占低 12 位，且 CPU 不基于 `flags`的信息来判断某项是页表项还是页目录项。要求回答中所有地址均使用用十六进制表示，如 `0x1edc8000`。

假定虚拟地址 0x8af4b000 映射到物理地址 0x01572000，则位于虚拟地址 `_____ ___` 的页表项内容应该是 `0x01572000 | flags`。位于 0xff1bc500 的页表项，它指向的页的虚拟地址起始于 `_____ ___`。

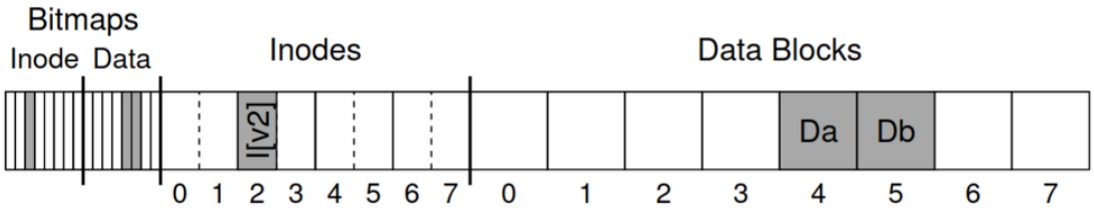
十一、文件系统 （共 5 题 共 10 分）

25.（2 分）

设一个简单的文件系统磁盘布局如下图：



其中 Bitmap 分为两部分，分别表示索引节点 Inode 和数据块的占用情况。Inode 存储文件的元数据，而实际数据则保存在元数据指向的数据块中。某一时刻需要对 Inode ID=2 的文件进行修改（简称 fd_add 文件操作），修改之后的磁盘布局变化如下：

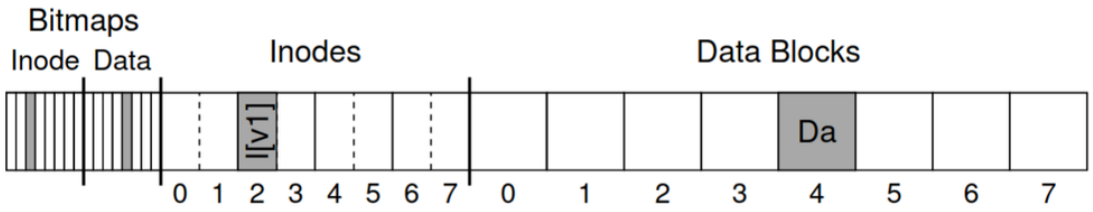


如上图所示：给该文件分配并修改了一个新的数据块 Db ，对应的数据块 Bitmap 需要被设置，该文件的 INode 也被修改。将三个修改分别记为 Db, B[v2], I[v2]。只有三个修改正确完成，此次 fd_add 文件操作才告成功。

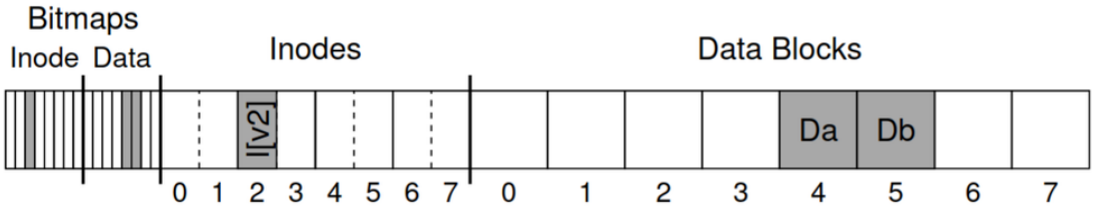
如果三个修改没有全部正确完成，系统就崩溃退出，那么会产生哪些后果（选择两种情况说明即可）？（回答字数 150 字）

26.（2 分）

设一个简单的文件系统磁盘布局如下图：



其中 Bitmap 分为两部分，分别表示索引节点 Inode 和数据块的占用情况。Inode 存储文件的元数据，而实际数据则保存在元数据指向的数据块中。某一时刻需要对 Inode ID=2 的文件进行修改（简称 fd_add 文件操作），修改之后的磁盘布局变化如下：



如上图所示：给该文件分配并修改了一个新的数据块 Db ，对应的数据块 Bitmap 需要被设置，该文件的 INode 也被修改。将三个修改分别记为 Db, B[v2], I[v2]。只有三个修改正确完成，此次 fd_add 文件操作才告成功。

为提高可靠性，可使用简单的日志系统提供的应对异常的可靠文件处理功能，在将数据实际写入磁盘之前要先将操作写入磁盘的日志区域。对于本题中的 fd_add 文件操作，请基于 Data Journaling，利用下述提供的选项给出完整的操作顺序（即由各种选项构成的有限有序列表，如可表示为 abBCef...），包括日志写入和将修改实际写入到磁盘。（注：a~e 中的写入操作是异步的，只是将操作提交到磁盘，并不等待操作完成后返回，而是立即返回，因此需要利用 f 的屏障操作进行同步；a~f 均有可能多次使用）

a: 将一条日志的开头写入日志区域；

b: 将修改 Db 写入日志区域；

B: 将修改 Db 实际写入磁盘；

c: 将修改 B[v2] 写入日志区域；

C: 将修改 B[v2] 实际写入磁盘；

d: 将修改 I[v2] 写入日志区域；

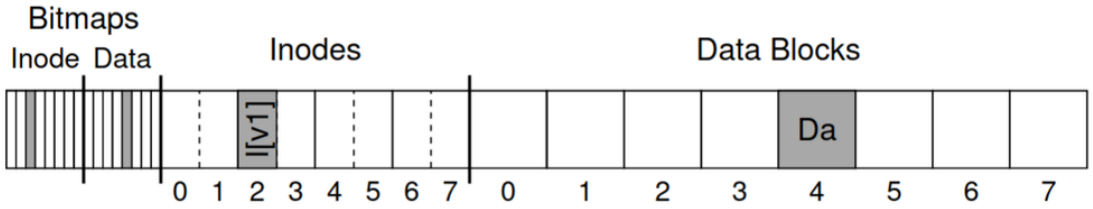
D: 将修改 I[v2] 实际写入磁盘；

e: 将一条日志的结尾写入日志区域；

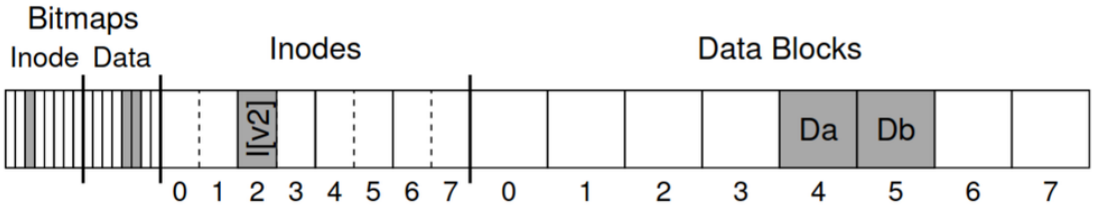
f: 等待之前的写入操作全部完成再继续操作。

27.（2 分）

设一个简单的文件系统磁盘布局如下图：



其中 Bitmap 分为两部分，分别表示索引节点 Inode 和数据块的占用情况。Inode 存储文件的元数据，而实际数据则保存在元数据指向的数据块中。某一时刻需要对 Inode ID=2 的文件进行修改（简称 fd_add 文件操作），修改之后的磁盘布局变化如下：



如上图所示：给该文件分配并修改了一个新的数据块 Db ，对应的数据块 Bitmap 需要被设置，该文件的 INode 也被修改。将三个修改分别记为 Db, B[v2], I[v2]。只有三个修改正确完成，此次 fd_add 文件操作才告成功。

为提高可靠性，可使用简单的日志系统提供的应对异常的可靠文件处理功能，在将数据实际写入磁盘之前要先将操作写入磁盘的日志区域。对于本题中的 fd_add 文件操作，请基于 Metadata Journaling，利用下述提供的选项给出完整的操作顺序（即由各种选项构成的有限有序列表，如可表示为 abBCef...），包括日志写入和将修改实际写入到磁盘。（注：a~e

中的写入操作是异步的，只是将操作提交到磁盘，并不等待操作完成后返回，而是立即返回，因此需要利用 f 的屏障操作进行同步；a~f 均有可能多次使用）

a：将一条日志的开头写入日志区域；

b：将修改 Db 写入日志区域；

B：将修改 Db 实际写入磁盘；

c：将修改 B[v2] 写入日志区域；

C：将修改 B[v2] 实际写入磁盘；

d：将修改 I[v2] 写入日志区域；

D：将修改 I[v2] 实际写入磁盘；

e：将一条日志的结尾写入日志区域；

f：等待之前的写入操作全部完成再继续操作。

28.（2 分）试从性能/可靠性/实现复杂度的角度来分析比较 Metadata Journaling 和 Data Journaling 方法的优劣之处。（回答字数 200 字）

29.（2 分）如何在不降低文件系统可靠性的前提下，进一步提高该文件系统的性能（给出两点即可）？（回答字数 150 字）