清华大学本科生考试试题纸	
考试课程：操作系统（A卷）	时间：2021年05月17日上午09:50~11:50

系别：_____	班级：_____	学号：_____	姓名：_____

答卷注意事项:	1.答题前在试题纸和答卷本上写明A卷或B卷、系别、班级、学号和姓名。
	2.在答卷本上答题时, 要写明题号, 不必抄题。
	3.答题时, 要书写清楚和整洁。
	4.请注意回答所有试题。本试卷有19个小题，共8页。
	5.考试完毕, 必须将试题纸和答卷本一起交回。

一、(22分)判断题

1. [ ] 在多CPU计算机系统中，操作系统可以通过禁用硬件中断来实现同步和互斥。
2. [ ] 临界区的访问规则包括忙则等待、空闲则入、有限等待。
3. [ ] 出现死锁的必要条件包括互斥、持有并等待、非抢占、按序等待。
4. [ ] 信号是一种进程间的软件中断通知和处理机制，其传输的信息是信号量。
5. [ ] 消息队列是由操作系统维护的以字节序列为基本单位的间接进程通信机制。
6. [ ] 共享内存是把同一个物理内存区域同时映射到多个进程的地址空间的直接进程通信机制。
7. [ ] 默认情况下的用户进程缺省打开的两个文件描述符（0、1）分别是(标准输入、标准输出)。
8. [ ] 目录是一种特殊的文件，其内容主要是由<文件名, 指向文件元数据的指针>项组成的列表。
9. [ ] 可用性高的系统其可靠性一定也高，反之亦然。
10. [ ] RISC-V的中断委托机制可以有选择地将某些中断交给S模式中的软件进行处理。
11. [ ] 两个线程可以访问同一个栈上的变量。

二、(78分)简答题

12. (10分)

针对RISC-V的硬件中断相关机制，请回答如下问题：

1. ）RISC-V的计算机中的Timer中断、software中断，串口中断、磁盘中断分别是属于Local Interrupts类中断还是Global Interrupts类中断?
2. ）Platform-Level Interrupt Controller (PLIC)具有什么作用？描述字数小于50字。
3. ）产生中断后，RISC-V硬件会把被中断的指令的 PC 保存在内核栈中还是保存在某CSR寄存器中？此时RISC-V硬件是否会屏蔽（禁用）中断？然后RISC-V硬件会根据什么信息的指引，跳转到哪里去？描述字数小于100字。

13. (12分)

设磁盘的初始磁头位置为53， 磁盘访问序列为：98, 183, 37, 122, 14, 124, 65, 67

1. ）请描述FIFO算法，最短服务时间优先 (SSTF)算法，SCAN算法的基本思路。描述字数小于150字。
2. ）请计算出采用这三种算法的磁头合计移动距离，请写出计算过程。

14. (14分)

对于读文件或接收UDP网络数据的用户进程而言，在操作系统的支持下，可以采用阻塞 I/O、非阻塞 I/O、多路复用 I/O、信号驱动 I/O、异步 I/O等I/O执行模型。请回答下列问题：

1. ）描述这些I/O执行模型特点分别是什么。描述字数小于200字。
2. ）对于实验指导chapter 7实现的操作系统中，发出read系统调用读取某磁盘文件内容的用户进程采用的是哪种I/O执行模型?
3. ）请用文字形式简要说明在实验指导chapter 7实现的操作系统中，用户进程发出read系统调用读取某磁盘文件内容后，操作系统内核完成此系统调用的整个执行过程（如果涉及到特权级切换，中断/异常处理，系统调用处理，中断/进程上下文切换，内存管理，进程调度，同步互斥，IO缓存处理，设备驱动等内核模块，请按内核执行顺序简要说明这些模块的执行过程，对涉及的每个内核模块执行过程的描述字数小于30字）。

注：对于第二小问和第三小问，请说明你的分析说明对象是ucore tutorial还是rcore tutorial

15. (6分)

实验框架中，用户态传入的字符串或者结构体的虚存地址作为系统调用的参数时，会通过translated_byte_buffer (rust框架) 或者copyin(C框架)进行拷贝。

1. ）请说明为何不直接把传入地址通过页表转换为物理地址对结构体或字符串进行访问。
2. ）请简要说明translated_byte_buffer或者copyin函数的实现方法。
3. ）如果一个用户进程的页表映射范围包括该用户进程和内核所在的全部地址空间，那么进程从用户态进入内核态后，内核是否可以直接基于用户态传入的虚拟地址对结构体或字符串进行访问？请简要说明理由。

16. (8分)

假定系统中的四种资源总量为Total = (5, 2, 3, 3)。 现在4个进程，它们的已分配资源情况如下。

进程名	资源A	资源B	资源C	资源D
P	2	0	1	1
Q	0	1	0	0
R	1	0	1	1
S	1	1	0	1
合计	4	2	2	3

各进程的未来资源需求情况如下。

进程名	资源A	资源B	资源C	资源D
P	1	1	0	0
Q	0	1	1	2
R	2	1	0	0
S	0	0	1	0

当前的可用资源Available=(1, 0, 2, 0)。

请问在当前状态下，如下的资源分配申请是安全的吗?

注：下面四个请求独立，没关系，没有先后顺序

1. ）进程P申请资源(1, 0, 0, 0)
2. ）进程R申请资源(1, 0, 0, 0)
3. ）进程Q申请资源(0, 0, 1, 0)
4. ）进程S申请资源(0, 0, 1, 0)

17. (8分)

小张写了个程序，希望执行程序中的三个线程，能够显示相关信息并正常结束：

```
sem_t semaphore;
pthread_mutex_t lock;
```

```
void threadA() {
  sem_wait(&semaphore); // line 1
  printf("threadA!");
  sem_post(&semaphore);
}

void threadB() {
  sem_wait(&semaphore); // line 1
  pthread_mutex_lock(&lock); // line 2
  printf("threadB!");
  pthread_mutex_unlock(&lock);
  sem_post(&semaphore);
}

void threadC() {
  sem_wait(&semaphore); // line 1
  pthread_mutex_lock(&lock); // line 2
  sem_wait(&semaphore); // line3
  printf("threadC!");
  sem_post(&semaphore);
  pthread_mutex_unlock(&lock);
  sem_post(&semaphore);
}

int main() {
  sem_init(&semaphore, 0, 2); // 初始化semaphore值为2
  pthread_mutex_init(&lock, NULL); //初始化互斥锁，信号量初始值为1
  phtread_t threads[3];  //线程标识符的数组
  pthread_create(&thread[0], NULL, threadA, NULL);
  pthread_create(&thread[1], NULL, threadB, NULL);
  pthread_create(&thread[2], NULL, threadC, NULL);
}
```

备注：

```
sem_t为资源信号量类型；
sem_wait函数等价于P操作，sem_post函数等价于V操作；
pthread_mutex_t为用于互斥操作的二值信号量（也称二进制信号量，互斥锁）类型；
pthread_mutex_lock函数对应互斥锁的lock操作，如果被锁住，该线程会被阻塞；
pthread_mutex_unlock函数对应互斥锁的unlock操作。

sem_init //初始化信号量的函数
第一个参数为指向信号量的指针
```

```
第二个参数值为 0，表示信号量被进程内的线程共享，且放置在此进程中所有线程都可见的地址上
第三个参数指定信号量的初始值

pthread_create //创建就绪态线程的函数
第一个参数为指向线程标识符的指针
第二个参数用来设置线程属性
第三个参数是线程运行函数的起始地址
最后一个参数是运行函数的参数

pthread_mutex_init //初始化互斥锁的函数
第一个参数是指向互斥锁的指针
第二个参数是指向互斥锁属性的指针，如果为NULL，则使用默认的互斥锁属性，即可睡眠的互斥锁
```

1. ）上述程序中的三个线程在操作系统调度管理下以某种顺序执行时，有时会没有任何输出。请给出一种没有任何输出的执行顺序的情况。并简要说明出现的这种情况在操作系统里一般称为什么现象?

回答的格式举例：

```
threadA: line1
threadC: line1
threadC: line2
threadC: line3
threadB: line1
threadB: line2
或简写为
A:1  C:1  C:2 C:3 B:1 B:2
```

2. ）为了避免“程序没有任何输出”的情况，让程序能输出并正常结束，小张想修改程序，使用下面的pthread_mutex_lock_safe替代pthread_mutex_lock，请帮助小张用类C代码补充下面的代码实现。请简要说明为何你的补充代码能够避免出现上一小问中的现象。说明字数小于50字。

```
void pthread_mutex_lock_safe() {
  _____A_____;
  pthread_mutex_lock(&lock);
  _____B_____;
}
```

18. (10分)

1. ）在 lab7 中我们专门为磁盘块建立了I/O缓存（磁盘块缓存），这样做有什么好处？在出现OS崩溃或断电情况，会带来什么问题?
2. ）I/O缓存会占用内存空间，如果操作系统中的内存空间紧张，是否可以把I/O缓存都释放掉来缓解内存空间紧张的情况？请简要说明理由。说明字数小于30字。
3. ）如果释放一部分I/O缓存就能缓解内存紧张情况，那应该基于什么原则考虑来选择释放哪些I/O缓存？描述字数小于30字。

19. (10分)

1. ）在类似ext3/4这样的文件系统中的日志机制的主要目的是什么?
2. ）日志文件系统（如ext3/4）与非日志文件系统（如fatfs, ext/ext2）相比，请分别说明在读文件操作和写文件操作上是否有不同？如有，请简要说明两种文件系统的操作的不同之处。说明字数小于50字。
3. ）某一个simple-journal-fs日志文件系统中日志模块的核心伪代码如下，请回答相关问题。

注：在本题中，journal, log 的中文含义都称为日志

相关数据结构定义和主要函数：

```
struct buf {
    int block_id;      // 数据所在的磁盘块
    char data[BLOCK_SIZE];  // 实际数据
}

struct membuf {
    int len;         // 有效 buf 的数量
    struct buf bufs[MAX_BUF_NUM];
} MEM_BUF;  // 全局 in memory buf

struct loghdr {
    int len;         // 有效 block_id 的数量
    int block_id[MAX_BUF_NUM];
}

// 将 data 写入第 block_id 个磁盘块，但仅会写入磁盘的非持久化缓存，需要 disk_flush() 完成实际写入。
void disk_write(int block_id, char* data);

// 从第 block_id 个磁盘块读出 data。
char* disk_read(int block_id);

// 通知磁盘将磁盘缓存中的数据完成实际写入。
```

```
void disk_flush();
```

将 log 和数据写入磁盘的相关函数：

```
struct loghdr* log_build() {
    if MEM_BUF->len == 0 {
        return NULL;
    }
    loghdr *hdr = empty_hdr();
    for(int i = 0; i < MEM_BUF->len; ++i) {
        // 对于将所有 memory buf 的数据写入从 LOG_BASE 开始的块
        disk_write(LOG_BASE + i, MEM_BUF->bufs[i].data);
        // 将对应的 block_id 写入 loghdr
        hdr->block_id[i] = MEM_BUF->bufs[i].block_id;
    }
    hdr->len = MEM_BUF->len;
    return hdr;
}

void data_apply() {
    for(int i = 0; i < MEM_BUF->len; ++i) {
        // 将 memory buf 的数据写入磁盘
        disk_write(MEM_BUF->bufs[i].block_id, MEM_BUF->bufs[i].data);
    }
}

int log_commit() {
    loghdr *hdr = log_build();
    if hdr == NULL {
        return 0;
    }
    disk_flush();
    disk_write(LOG_HEADER, (char*)hdr);
    disk_flush();
    data_apply();
    disk_flush();
    hdr->len = 0;
    disk_write(LOG_HEADER, (char*)hdr);
    disk_flush();
    MEM_BUF.clear();
    return 0;
}

```

```
40 int log_recover() {
41     loghdr *hdr = disk_read(LOG_HEADER);
42     for(int i = 0; i < hdr->len; ++i) {
43         char* data = disk_read(LOG_BASE + i);
44         // YOUR CODE, 仅有一行
45     }
46     disk_flush();
47     hdr->len = 0;
48     disk_write(LOG_HEADER, (char*)hdr);
49     disk_flush();
50     return 0;
51 }
```

4. ）在 `log_commit` 函数中，以下操作各对应哪一行代码（写行号即可，已知都不是 `disk_flush()` ）？
    - Journal write
    - Journal commit
    - Data write
    - Journal free
5. ）请补全 `log_recover` 中第 44 行空缺的代码
6. ）小张是个冒失鬼，他不小心把第 35 行的 `disk_flush` 注释掉了，请问是否可能会导致错误？如果不会导致错误，请解释原因。如果会导致错误，请说明何种情形下会导致错误?
7. ）ext4 文件系统在缺省情况下，日志只记录元数据（metadata），其磁盘写入顺序为：

- Data write
- Journal metadata write
- Journal commit
- Metadata write
- Journal free

这样做与上述simple-journal-fs日志文件系统的做法相比，有何好处？为何先写入 data ?