

清华大学本科生考试试题专用纸

考试课程：操作系统（A 卷）

时间：2018 年 05 月 25 日下午 12:45~15:05

系别：_____ 班级：_____ 学号：_____ 姓名：_____

- 答卷注意事项：
1. 答题前，请先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 29 个题目，共 5 页。
 5. 考试完毕，必须将试题纸和答卷本一起交回。

一、对错题（15 分）

注意：回答请用 V 表示正确，用 X 表示错误：

1. ☐ 在多CPU场景下，多个线程通过自旋锁(spinlock)争抢进入临界区执行，第一个成功进入临界区的线程是第一个执行自旋锁争抢的线程。
2. ☐ 运行在内核态的内核线程共享操作系统内核态中的一个页表。
3. ☐ 操作系统创建用户进程时需要为此用户进程创建一个内核栈用于执行系统调用服务等。
4. ☐ 通用操作系统的调度算法的主要目标是低延迟，高吞吐量，公平，负载均衡。
5. ☐ 单处理器场景下，短剩余时间优先调度算法(SRT)可达到具有最小平均周转时间的效果。
6. ☐ 单处理器场景下，无法通过打开和关闭中断的机制来保证内核中临界区代码的互斥性。
7. ☐ 信号量可用于解决需要互斥和同步需求的问题。
8. ☐ 属于管程范围的函数/子程序相互之间具有互斥性。
9. ☐ 操作系统处于安全状态，一定没有死锁；操作系统处于不安全状态，可能出现死锁。
10. ☐ 80386取指地址是base+eip，base是隐藏寄存器，初始化为0xffffffff，eip初始化为0xfffff，故执行的第一条指令是0xfffffffff。
11. ☐ 在x86-32 CPU下，操作系统可以实现让用户态程序直接接收并处理硬件中断。
12. ☐ 由于符号链接（软链接）实际上是一类特殊的文件，它的内容就是其所指向的文件或目录的路径，所以符号链接可以指向一个不存在的文件或目录。
13. ☐ 文件系统中，用于存储“文件访问控制信息”的合理位置是文件分配表。
14. ☐ 在操作系统中一旦出现死锁，所有进程都不能运行。
15. ☐ 在ucore for x86-32中，子进程通过sys_exit()执行进程退出时，ucore kernel会先释放子进程自身内核堆栈和进程控制块等，再唤醒父进程（或initproc），最后执行iret返回。

二、填空题（30 分）

小强同学认真上课听讲，参与讨论，并完成了从lab0~lab8的所有实验，在学习过程中，了解和学到了很多知识。下面是他的学习心得，请补充完整。

16. 小强发现完成实验需要在Linux下操作很多命令行工具，于是他认真学习了lab0中的知识，了解到git的强大版本管理功能，Linux中在命令行模式下可以通过执行一条命令“(__16.1__)”

https://github.com/chyyuu/ucore_lab.git”来首次获得整个实验的代码。如果编写完实验内容，可通过执行一条命令“(__16.2__)”来完成整个lab的编译和执行代码生成。

17. 在完成lab1的过程中，了解到在80386保护模式下，如果产生了外部中断，CPU需要开始保存当前被打断的执行现场，以便于将来恢复被打断的程序继续执行。这需要利用栈来保存相关现场信息，即依次压入当前被打断控制流涉及到的(__17.1__)、(__17.2__)、(__17.3__)等具体的硬件信息。
18. 在完成lab2的过程中，需要了解x86-32的内存大小与布局，页机制，页表结构等。硬件模拟器提供了128MB的物理内存，如ucore kernel需要通过页表管理整个128MB的物理内存，则页表总共需要占用(__18.1__)KB的内存空间。
19. 在完成lab3的过程中，ucore操作系统在页机制基础上，并利用异常机制建立了虚存管理策略与机制。在产生页面访问错误异常时，CPU直接把表示页访问异常类型的值（简称页访问异常错误码，errorCode）保存在(__19.1__)中。ucore通过 x86-32 CPU 中的(__19.2__)寄存器可以获得发生页面访问错误异常时的线性地址。
20. 在完成lab4/5的过程中，了解到操作系统管理内核线程或用户进程的一个关键数据结构是(__20.1__)，其中包含了(__20.2__)，用于进程/线程切换涉及的保存与恢复进程/线程上下文，还包含了(__20.3__)，用于用户态/特权态切换涉及被中断/异常打断的执行上下文。
21. 在完成lab6的过程中，小强发现执行测试时调度过程的显示结果很不稳定，通过与同学交流，发现是由于自己在windows中建立了一个virtualbox虚拟机环境，在virtualbox虚拟机环境下，再执行qemu模拟器导致ucore的实际上执行时间变动很大。为减少这种变动，小强采取了(__21.1__)【不超过20个字】的方法，改善了此问题，并取得了稳定的实验结果。
22. 在完成lab7的过程中，小强发现本实验主要通过建立(__22.1__)机制来完成信号量机制，主要通过建立(__22.2__)机制来完成管程机制，且此管程的实现采用的是Mesa、Hoare、Brinch Hanson 三种语义方式中的(__22.3__)语义方式。
23. 在完成lab8的过程中，小强发现ucore设计了文件系统抽象层--(__23.1__)，它提供一个统一的文件系统操作界面和编程接口，可支持不同的具体文件系统。在对具体文件系统SFS的分析过程中，小强发现SFS在硬盘上的主要内容包括(__23.2__)，记录了所在硬盘的扇区总数量和未用扇区数量；还有(__23.3__)，记录了已用扇区和未用扇区的位置；(__23.4__)记录了，根目录的内容。
24. 现有一个RAID磁盘阵列，包含6个磁盘，每个磁盘大小都是2TB，最大写入速度 200 MB/s，最大读取速度 250 MB/s 的硬盘。用它们分别组成RAID级分别为0、1和5。假设在理想情况下（无中断、异常、预先缓存等外在干扰因素等），请回答下列问题。
 - A) 用它们组成的RAID0阵列的总可用空间为 (__24.1__)，最大写入速度为 (__24.2__)，最大读取速度为 (__24.3__)；
 - B) 用它们组成的RAID1阵列的总可用空间为 (__24.4__)，最大写入速度为 (__24.5__)，最大读取速度为 (__24.6__)；
 - C) 用它们组成的RAID5阵列的总可用空间为 (__24.7__)，最大写入速度为 (__24.8__)，最大读取速度为 (__24.9__)。
25. 假定在X86-32平台上的ucore的虚拟存储系统中，采用4KB页大小和二级页表结构。请补全功能为通过虚拟地址找到对应的页表项的get_pte()函数。

可能需要用到的函数有：

page2pa() 获取物理页对应的物理地址；

page2ppn() 获取物理页对应的物理页号；

pa2page() 获取物理页号对应的物理页数据结构指针；

page2kva() 获取物理页对应内核虚拟地址；

kva2page() 从内核虚拟地址获取物理页数据结构指针；

可能用到的宏有：

memset(p,v,n) 对指定地址p开始的长度为n的内存区域进行赋值v

PDX(la) 虚拟地址la对应的页目录项序号；


KADDR(pa) 物理地址pa对应的内核虚拟地址；

PADDR(kva) 虚拟地址kva对应的物理地址；

PTE_P: 存在标志位

PTE_W: 可修改标志位

PTE_U: 用户可访问标志位

```
pte_t *get_pte(pde_t *pgdir, uintptr_t la, bool create){
    pde_t *pdep = __25.1__;
    if(!*pdep & __25.2__){ 
        struct Page *page;
        if(!create || (page = alloc()) == NULL){
            return NULL;
        }
        set_page_ref(page,1);
        uintptr_t pa = __25.3__;
        memset(__25.4__,0, PGSIZE);
        *pdep = __25.5__;
    }
    return &((pte_t *)KADDR(PDE_ADDR(*pdep)))[PTX(la)];
}
```

三、问答题

26. （18分）假定某文件系统采用多级索引分配的方法，普通文件的索引节点(inode)中包括一个占8字节的文件长度字段和15个占4字节的数据块指针（即块号）。其中，前12个是直接索引块（direct block）的指针，第13个是1级间接索引块（indirect block）指针，第14个是2级间接索引块（doubly indirect block）指针，第15个是3级间接索引块（triply indirect block）指针。间接索引块中连续存放占4字节的数据块指针。数据块以及间接块的大小为4KB。请回答下列问题。
- A) 计算该文件系统理论上能够支持的单个文件最大长度。
 - B) 假设读取磁盘上一个块需要1ms，块缓存机制只缓存文件的索引节点，并且所有需要的索引节点都已加载到缓存；不缓存数据块以及间接索引块；读取文件操作不会发生写入操作，访问内存的时间忽略不计。计算从头到尾读取一个8MB（即2048块）文件需要的时间。

C) 假设读取磁盘上一个块需要1ms，块缓存机制缓存文件的索引节点、数据块和间接索引块，并且所有需要的索引节点都已加载到缓存；开始时缓存没有加载任何数据块或者间接块；读取文件操作不会发生写入操作，访问内存的时间忽略不计。计算从头到尾读取一个8MB（即2048块）文件需要的时间。

27. （5分）无锁 (Lock-Free) 数据结构在工程实践中有十分重要的应用，因而常用处理器的指令集都提供了相应的指令来帮助我们实现无锁数据结构，如下为 x86 平台的 cmpxchg 指令的伪代码。

// Pseudocode for CMPXCHG instruction. It executes atomically.

```
int cmpxchg(void* addr, uint32_t oldval, uint32_t newval) {
    if (*addr != oldval) {
        return 0;
    }
    *addr = newval;
    return 1;
}
```

考虑使用该指令实现一个无锁 LIFO 队列

```
struct node {
    struct node* next;
    /* Other data fields */
};
struct node* head = NULL;

void push(struct node* node) {
    do {
        node->next = head;
    } while (!cmpxchg(&head, (uint32_t)node->next, (uint32_t)node));
}

struct node* pop() {
    while (1) {
        struct node* node = head;
        if (node == NULL) {
            return NULL;
        }
        struct node* next = node->next; // (*)
        if (cmpxchg(&head, (uint32_t)node, (uint32_t)next)) {
            return node;
        }
    }
}
```

上述实现存在 bug，假设现在有 CPU1 和 CPU2 同时操作该 LIFO 队列，请给出一个操作序列，使得最终该队列处于一个不一致的状态，即已经弹出的元素仍然在 LIFO 队列中或者未弹出的元素

不在 LIFO 队列中。以下给出了 LIFO 队列的初始状态和第一个操作，请补全能够触发 bug 的操作序列

o) LIFO 队列初始状态: head -> node A -> node B -> node C -> NULL

1) CPU₁ 开始调用 pop, 运行至函数体内 (*) 处, 此时有 node = A 和 next = B

2)

28. (20分) 理发店理有m位理发师、m把理发椅和n把供等候理发的顾客坐的椅子。理发师按如下规则理发。

1) 理发师为一位顾客理完发后, 查看是否有顾客等待, 如有则唤醒一位为其理发; 如果没有顾客, 理发师便在理发椅上睡觉。

2) 一个新顾客到来时, 首先查看理发师在干什么, 如果理发师在理发椅上睡觉, 他必须叫醒理发师, 然后理发师给顾客理发; 如果理发师正在理发, 则新顾客会在有空椅子可坐时坐下来等待, 否则就会离开。

请回答如下问题:

A) 用管程机制实现理发师问题的正确且高效的同步与互斥活动; 要求用类C语言的伪代码实现, 并给出必要的简明代码注释。

B) 请按理发规则的要求, 给出测试用例; 要求至少给出5种可能情况的测试用例。

29. (12分) 设lab6中使用Stride调度算法, 取BIGSTRIDE=100, 假定各个进程的stride初始化为0。

(注: lab6中的stride (32位整数, 当前总共走了多少) 和pass (32位整数, 每一次走多少 pass=BIGSTRIDE/priority, 100>priority>1) 的含义和论文原文含义相反) 请回答下列问题。

A) 如果不考虑进程stride的值的溢出, 那么对于任意两个进程A、B的stride值SA和SB, 应当恒有 $\text{abs}(SA-SB) \leq (\underline{\hspace{1cm}}1\underline{\hspace{1cm}})$, 为什么?

B) 考虑到 $\text{abs}(SA-SB)$ 这一性质, 假设stride值存在溢出, 可将stride值的更新变为:

$\text{stride} = (\text{stride} + \text{BIGSTRIDE}/\text{priority}) \bmod n$

那么, 只要 $n > (\underline{\hspace{1cm}}2\underline{\hspace{1cm}})$, 那么stride算法就可以正常运行, 为什么?