

第五章 线程与进程

概念题

- 线程: 程序在虚拟CPU上的执行过程,或者说执行中的程序.
- 进程:一组线程以及这组线程所依赖的虚拟运行环境(CPU,内存,设备).
- **并发执行:** 多个虚拟CPU,逻辑上连续执行,但是实际上交替执行.
- 线程控制块(TCB): 操作系统描述和管理线程的数据结构(标识,状态,上下文)
- 进程控制块(PCB): 操作系统描述和管理进程的数据结构.包含(TCB,资源描述,资源映射)
- 虚拟运行环境:地址空间(数据段,代码段,堆栈)和可用的设备.
- 内核级线程:由操作系统进行管理的线程.
- 共享内存:多个进程都可以访问各自的虚拟内存区域,进程虚拟地址映射到相同的物理地址.
- 消息传输:直接通信要显式指定对方;间接通信可以不同时在线.
- 吞吐量: 单位时间内完成的进程个数(宏观量)
- 响应时间:发出第一个请求,到接收到第一个响应,所花费的时间.
- 多线程: 一个程序中可以同时定义多个线程并同时运行它们,每个线程可以执行不同任务.

线程与进程

- 线程状态
新建.就绪.执行,阻塞,结束
- 操作系统对线程得操作:
创建,撤销,阻塞,调度,唤醒.
- 为什么说创建新进程的代价要远高于创建线程.
 - 创建新的进程付出创建虚拟运行环境的代价.
 - 创建新的线程只需要分配特定的资源
 - 进程上下文的切换代价更高,需要更改cache,TLB,内存

- **分类依据**

- 以进程为单位

操作系统进行设备和资源分配的基本单位

虚拟运行环境(地址空间(数据段,代码段,堆栈),可用的设备,,虚拟内存中的地址划分)

打开的文件是进程内线程共用

cache,TLB,内存等.

- 以线程为单位

CPU分配的基本单位

线程控制块,运行栈(不同线程之间不会共享栈)

- 内核级线程与用户级线程的区别
 1. OS管理/用户程序自己管理
 2. 可以抢占,切换代价高,只会阻塞自己,适用于多核,多CPU

非抢占式的绿色线程,I/O阻塞整个进程,不适用于多核,多CPU
- 进程的撤销?

进程内所有线程执行结束后,进程给其父进程发送信息,令父进程撤销当前进程,释放全部资源.
- 什么时候需要执行CPU调度算法?
 1. 当前线程执行结束
 2. 当前线程主动放弃CPU
 3. 当前线程阻塞

- 4. 就绪队列中出现高优先级的线程抢占
- 5. 时钟中断(时间片结束)
- 线程的CPU执行期预估
假设线程的首个CPU执行期为 s_0 ，实际值为 t_0
则下一个CPU执行期为 $s_1=a*t_0+(1-a)*s_0$ ，其中 $0 \leq a \leq 1$
依次类推，第n个CPU执行期可以估算为： $s_n=a*t_n+(1-a)*s_{n-1}$

CPU调度

调度方式分类？ 答：抢占式和非抢占调度

各自产生调度的情况？ 答：4,5和1,2,3

两者的比较？ 答：系统开销和及时性

就绪队列为空，进行什么操作？ 答：执行闲逛进程，不断调用 `yield()` 调用CPU调度器。

三级调度？ 答：作业调度，内存调度，进程调度

从哪些维度评价调度算法？

答：CPU利用率，吞吐量，等待时间，响应时间，周转时间。

平均等待时间，平均周转时间，平均带权周转时间。

列举七种CPU调度算法，并说明各自的优缺点。

提示：

FCFS, SJF, 高响应比, 优先级, RR, 多级队列, 多级反馈队列, CFS

公平,简单,无饥饿;平均等待时间最短;偏向于等待时间长和执行期短的,折中,不饥饿;响应速度快,用户体验好,开销大,平均周转时间增加;.....;兼顾多方面的系统目标,

上下文切换都是切换啥？ 答：CPU寄存器的保存与恢复, CPU cache的更新, 快表TLB的更新, 虚拟存储器开销。

进程通信的两种模式？ 答：消息传输, 共享内存。两者都通过通信接口和OS

作业

- 给出两个程序的例子，其中多线程的表现不如单线程的表现好？
 - 任何类型的顺序程序都不是线程化的好选择，一个例子是计算个人纳税申报表的程序。
 - 一个例子是“shell”程序，如 C-shell 或 Korn shell。这样的程序必须密切监视自己的工作空间，例如打开的文件、环境变量和当前工作目录。

王道

- 在单处理器系统中，任何时刻都只有一个进程处于运行态？

答：不对。因为存在着死锁的情况，在死锁发生时，没有进程处于运行态，都处于阻塞态。

- 并发进程失去封闭性，是指？

- 多个相对独立的进程以各自的速度向前推进
- 并发进程的执行结果与速度无关
- 并发进程执行时，在不同时刻发生的错误。
- 并发进程共享变量，其执行结果与速度有关

答：D（不设置同步机制，结果不唯一）

- 在引入线程的系统中，进程仍是资源调度和分派的基本单位？

看题目描述。**调度于分派** 是指CPU，显然线程才是资源调度于分派的基本单位。

- 全局赋值变量,常量值都存在于进程实体的正文段**;
- 未赋值的局部变量,函数调用的实参传递值都存在于进程实体的**栈段**;
- 动态分配的存储区,存储于进程实体的**堆段**;
- 进程的优先级存在于进程实体的**PCB**中.
- 系统动态DLL库中的系统线程,被不同的进程所调用,它们是()的线程.
 - A. 不同
 - B. 相同
 - C. 可能相同可能不同
 - D. 不能被调用

答: B 是相同的线程
- 以下描述中,()不是多线程系统的特长?
 - A. 利用线程并行地计算矩阵乘法
 - B. Web服务器利用线程响应HTTP请求
 - C. 键盘驱动程序为每个正在运行的应用分配一个线程,响应该应用地键盘输入
 - D. 基于GUI的调试程序用不同的线程分别处理用户输入,计算和跟踪等操作.

答: C(用户输入速度相较于CPU运算速度比较慢,用一个线程就可以处理)
- 管道通信,相关说法正确的是?()
 - A. 一个管道可以实现双向数据传输; B. 管道的容量仅受磁盘容量大小限制
 - C. 进程对管道进行读和写操作都有可能被阻塞
 - D. 一个管道只能有一个读进程或者一个写进程对其进行操作

答: C(A只能进行单向传输,一读一写,B管道存放在内存中,特殊类型的文件,大小通常为一页,D读写进程不唯一)
- 操作系统会为每一个用户级线程创建一个进程控制块?
- 不一定,仅在一对一模型下才成立.
- 多线程和多任务的区别?
 - 针对程序而言;针对操作系统而言;
 - 代表一个程序可以同时执行的线程个数;代表操作系统可以同时执行的程序个数
- 采用优先级进程调度时,运行进程是否一定是系统中优先级最高的进程?
- 不一定,可能在阻塞队列中等待被唤醒.
- 不能进行进程的调度与切换的情况:
 - 在处理终端的过程中
 - 其他需要完全屏蔽中断的原子操作过程

(在临界区实际上完全可以进行调度)
- 进程的优先级设置通常遵循下列三条:
 - 系统进程>用户进程
 - 交互进程>非交互进程
 - I/O型进程>计算型进程

第六章 内存管理

概念

- 连续内存分配:为作业分配连续的内存空间,将作业全部装入到内存中.
- 非连续内存分配:作业各部分在内存中不必相邻,一个作业可以分多次装入内存.
- 固定分区:OS事先将内存划分成多个区域,分区个数一般是固定的.每个分区在每个时刻只能被分配给一个作业.
- 动态分区:事先不对内存进行划分,而是根据作业的实际大小为其分配内存.
- 内碎片:**已经被分配出去(能明确指出属于哪个作业),但是不能被利用的内存空间.
- 外碎片:**尚未分配出去(不属于任何作业),但是由于太小而无法分配给任何作业的空闲分区,称为外碎片.

7. 段: 段是用户作业中, 具有逻辑意义的一部分
8. 页: 物理地址空间和虚拟地址空间按照同一尺度 (例如1kB) 进行分割, 大小必须为2的整数次幂
9. 快表 (TLB) : Translation Lookaside Buffer : CPU中设置的专用来存储页号与页框号映射的高速缓冲区
10. PTR(页表寄存器): Page Table Register
11. 请求调页: 执行一个进程之前, 仅将PCB装入内存, 程序和数据仅当访问到且没有装入内存时才装入, 成为缺页异常处理.
12. 缺页率: 缺页次数/访存次数;
13. ms和ns之间的转换率是 10^6
14. 局部性原理: 时间局部性和空间局部性
15. 时间局部性: 程序执行过程中, 如果某个信息项被访问, 那么近期它很有可能被再次访问
16. 空间局部性: 如果程序访问某个存储单元, 那么近期内很有可能会访问附近的其他存储单元
17. 局部: 进程访问的页的集合, 这些页在使用上具有关联性.
18. 局部模式: 进程的执行过程中, 其访问的内存空间从一个局部迁移到另一个局部, 这种运行方式被称为局部模式.
19. 工作集(Working Set): 进程在某个时刻t之前 Δ 次访问中所访问的页的集合, 记为WS(t, Δ), Δ 称为工作集的窗口
20. Belady异常: 分配到的页框数越多, 缺页率越高
先进先出的替换算法, 完全不考虑使用频率, 即使增加了实页数, 多贮存的部分接下来常访问可能性也不一定大, 也就并不一定能增加命中率。缺页率反而降低的原因在于, 把经常使用的页置换出去.
21. OPT算法(optimal): 最优置换
22. 虚拟内存技术: 进程执行时不必完全装入内存, 由系统实现对换功能.
23. 虚拟存储器: 通过采用虚拟内存技术, 系统为用户提供了一个比实际内存大得多的存储器, 成为虚拟存储器.(大小由计算机的地址结构决定, 并不是内存和外存的简单相加)
24. **抖动**(颠簸): 指如果分配给进程的存储块数量小于进程所需要的最小值, 进程的运行将很频繁地产生缺页中断, 这种频率非常高的页面置换现象称为抖动。

各种内存分配

	连续分配	纯分段	纯分页
外碎片	动态分区分配方式会产生外碎片	纯分段会产生外碎片	纯分页不会产生外碎片
内碎片	固定分区分配方式会产生内碎片	纯分段不会产生内碎片	纯分页会产生内碎片, 但是十分有限
进程间代码共享	不能实现进程间共享	能够进程间共享	能够进程间共享

固定分区的缺点

- 小作业占据大空间
- 大作业找不到合适的分区
- 分区个数固定, 限制了系统中进程的并发程度.
- 无法实现内存共享

连续内存分配的特点

- 优点

地址变换, 存储保护容易实现;

支持多道程序设计技术

- 缺点

无法实现内存共享
不支持虚拟存储技术
容易产生碎片
难以为大作业寻找合适的连续分区

段分配与连续分配的不同

分段以段为单位进行内存分配和管理。
连续内存分配是把整个作业看成一个整体进行内存分配。
(偏个题：每个段内的地址和连续分配一样，是连续的)

硬件相关

固定分区：基址寄存器和限长寄存器。
动态分区的机制与固定分区大致相同。
段分配：段基址寄存器，段长寄存器
页分配：有效位，读写执行位，存在位

分页和分段的比较

不同点：

1. 页由**系统划分**，对程序员透明
2. 页不具有**逻辑意义**
3. 页的**大小固定**
4. **宏观上没有碎片**，作业的最后一个页面可能产生碎片
5. 分页模式下，程序员访问存储单元，用的是逻辑地址空间中的一维地址。

共同点：

1. 作业在内存中不连续存访
2. 页的装入和地址变换过程和段非常相似

请求调页和纯分页的不同点

纯分页：一次装入全部作业的所有页

请求调页：

页表中增加的设计(存在位和脏位)+异常(缺页异常)

操作系统中的改变：页故障异常处理(页置换)

硬盘的改变：对换区

页故障的处理(6)

- ① 缺页异常，由MMU检测存在位引发
- ② 缺页异常处理，进入OS
- ③ 写回换出页，检测脏位
- ④ 读入换入页(最耗时)
- ⑤ 修改页表：页框号、存在位
- ⑥ 重新执行，PC不变

发生抖动的表现：

CPU利用率低，缺页率高；
也就是系统会花很多时间在磁盘的读写，而不是在CPU运行中。

降低缺页率的方法

1. 充分利用进程的局部性原理
2. 分配足够多的内存(页框数量)，以容纳当前进程的局部
3. 采用合适的置换算法(将进程的局部留在内存，不用的置换出去)

如何选择合适的工作集

Δ 的变化:如果WS中的页能全部装入内存,缺页率最小.当 $|WS|/\Delta$ 足够小,WS可以认为是当前进程的局部.

当 Δ 增加到某个值后,工作集相对稳定.

对t的变化不敏感=》当前的工作集可以作为将来近期的工作集

对 Δ 的变化也不敏感=》说明工作集已经包含了进程的一个局部

进程应得的内存数量:下限是局部.

页框分配方法:

固定分配方法:依据的是进程静态的特征,没有考虑动态需求

平均分配

按比例分配(按该进程页的数量与整体所有进程的页的数量比例)

按优先级分配

可变分配方法(根据进程的实际运行)

全局置换(缺页率高的进程占用缺页率低的进程的页框)

局部置换(在本进程中选择要置换的页)

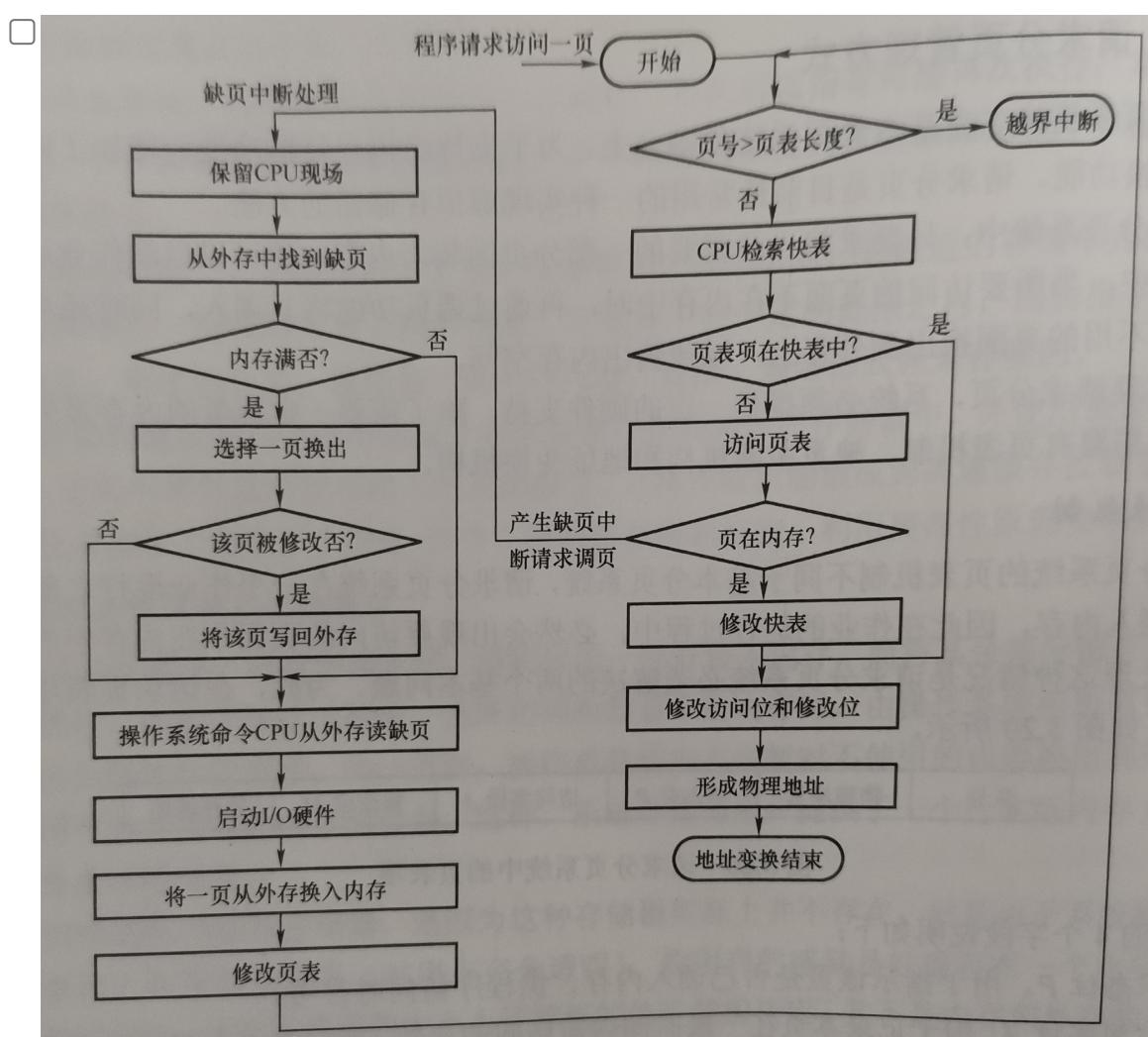
置换算法(目的:降低缺页率,保留进程的局部在内存中)

最优置:淘汰将来最久不用的页(理想算法,作为比较模板)

先进先出置换算法:置换在内存中驻留时间最长的页(存在 **Belady** 异常)

最久未用(LRU):least recently used.(利用局部性原理,与最优置换是对称的关系,但是需要记录内存中所有页最后一次被访问的时间;太耗时,且记录时间的字段有可能溢出)

CLOCK方法.(基于引用位的硬件算法,淘汰遇到的第一个引用位为0的页)



作业题

1. 使用分页的系统中,一个进程为什么不能访问不属于它的内存空间?

在分页系统中,不同的进程存在于不同的虚拟地址空间(通过页表进行描述),进程不能直接访问其他进程的资源.在硬件层次,地址转换由MMU完成,它在有非法地址的情况下产生异常,禁止进程非法访问不属于它的页框.

2. 操作系统怎样做到允许某个进程访问其他进程的空间?

进程之间可以通过共享内存完成通讯,比如说可重入代码通过共享页可以完成不同进程之间共享同一段代码.

3. 为什么分段和分页有时会在一个方案里同时被使用(为什么要段页式内存管理)?

(1) 纯分段的好处是实现了虚拟地址空间,且进程段实际上是一个逻辑单元,段与段之间存在语义.但是纯分段还是需要OS去进行较为复杂的内存管理,纯分段还会产生外碎片.

(2) 纯分页的好处是实现了虚拟地址空间,且由于页的大小固定且有限,在内存管理中简化了OS的负担.但是纯分页虽然很少,但是仍然会产生内碎片,且由于页的大小固定,页不再是一个逻辑单元.

(3) 纯分段与纯分页都具有一定的缺点,而采用段页式存储管理方式,在一定程度上具有纯分段和纯分页二者的优势,且弱化了缺点.

4. 发生颠簸时,下面哪种行为是有助于减轻颠簸/提高CPU利用率的?

- 1. 更快的CPU (x)
- 2. 安装更大的磁盘 (x)
- 3. 增加进程并发程度 (x)
- 4. 减少进程并发程度 (v)
- 5. 安装更多的主存 (v)
- 6. 安装更快的磁盘或者多控制器 (v)(不会减轻颠簸,但会提高CPU利用率,页错误处理时间减小)
- 7. 页置换策略中添加预取页(预约) (v)(能够使发生在这个进程上的系统颠簸程度降低)
- 8. 增大页大小 (不一定)(在两个页边界不断跨页访问的可以消除)

5. 使用请求调页需要什么硬件支持? 答:页表寄存器+转换表缓冲区(TLB)

6. 颠簸产生的原因是什么?如何检测颠簸?如何减轻颠簸?

- 1. 早期调页系统中,CPU如果利用率太低,OS就会引入新进程以增加多道程序的程度.如果采用全局置换算法,OS会置换页而不管这些页是属于哪些进程的.如果一个进程进入一个新的执行阶段,需要更多页,它开始产生页错误,并从其他进程那儿拿到页.然而其他的进程也需要这些页,所以它们也会产生页错误,从而又从其他的进程那儿拿页.这样彼此取走活跃的页,导致不断出现页错误,称为系统颠簸.
- 2. 当OS发现CPU的利用率突然断崖式下降.
- 3. 采用局部置换算法,保存进程的局部,减轻系统颠簸的发生.

7. 有没有可能同时拥有两个工作集?一个代表数据而一个代表代码?

可以,比如在哈佛架构的计算机中,代码和数据自然是分开的,此时CPU对代码和数据的访问可以分成两个工作集,代码本身由于循环与子程序就拥有局部性,而数据由于数组等顺序结构的存在也具有一定的局部.工作集模型在能够比较好地描述这些局部特征.

王道题

1. 在使用交换技术时,若一个进程____则不能交换出内存.

- A. 创建
- B. IO操作
- C. 处于临界段
- D. 死锁

答:(B,进行IO操作时,该进程不一定处于空闲状态,而是在不停的读取数据进入内存中,因此不可交换出内存;

但是当IO操作通过高速缓冲区完成时,可以交换出内存)

2. 采用分页或者分段管理后,提供给用户的物理地址空间____

- A. 分页更多
- B. 分段更多
- C. 不能确定

答:(C,因为提供的物理地址空间=总空间-页/段表长度)

3. 可重入程序是通过()方法来改善系统性能的

- A. 改变时间片长度
- B. 改变用户数
- C. 提高对换速度
- D. 减少对换数量

答:D(通过共享来使用同一块存储空间,从而减少对换)

4. 操作系统实现()存储管理的代价最小

- A. 分区
- B. 分页
- C. 分段
- D. 段页式

答:A (后三种都需要数据结构(页表/段表的支持以及硬件地址转换机构))

5. 某个操作系统对内存的管理采用页式存储管理方法,所划分的页面大小()

- A. 根据内存大小确定
- B. 必须相同
- C. 根据CPU的地址结构确定
- D. 根据外存和内存的大小确定

答:(B,如果不相同,没办法通过一维地址和除法确定其页号;划分页面大小由 **进程平均大小**, **页表所用长度** 等多个因素确定)

6. 引入段式存储管理方式,主要是为了更好地满足用户的一系列要求,下面选项中不属于这一系列要求的是?

- A. 方便操作
- B. 方便编程
- C. 方便共享和保护
- D. 动态链接和增长

答:A(剩下三条都是)

7. 对主存储器的访问,()

- A. 以块(即页)或段为单位
- B. 以字节或字为单位
- C. 随存储器的管理方案不同而异
- D. 以用户的逻辑记录为单位

答:B(是**访问单位**而不是**分配单位**)

8. 分页式存储管理,要求()

答: 每个进程拥有一张页表,且进程的页表驻留在内存中.

多个进程并发执行的时候,所有进程的页表大多数驻留在内存中,且系统中只设置一个页表寄存器.

进程未执行,页表起始地址和页表长度存放在本进程的PCB中,当被调度时,将数据装入页表寄存器.每个进程都有一个单独的逻辑地址,有一张属于自己的页表.

9. 传统存储管理方式的特征

一次性(必须一次性全部装入内存中)

驻留性(作业被装入内存后一直驻留直到作业结束)

10. 虚拟存储器的特征

多次性(不必一次性全部装入)

对换性(无需在作业运行时一直常驻内存)

虚拟性(从逻辑上扩充内存的容量)

11. 考虑页面置换算法,系统有m个物理块供调度,初始时全部为空,页面引用串长度为p,包含了n个不同的页号,无论用什么算法,缺页次数不会少于()

- A. m
- B. n
- C. p
- D. $\min(m,n)$

答:B

12. 导致LRU算法实现起来耗费高的原因是()

答:需要对所有的页进行**排序**.导致的后果是,需要硬件的特殊支持.

13. 请求分页式存储管理的主要特点是()

- A. 消除了页内零头
- B. 扩充了内存
- C. 便于动态链接
- D. 便于信息共享

答:(B)

14. 下列措施中,能够加快虚实地址转换的是()

- A. 增大快表容量
- B. 让页表常驻内存
- C. 增大交换区

答:A,B

第七章 输入输出(设备管理)



概念

1. 设备:连接CPU的所有设备,被称为输入输出设备,简称外设.
2. 逻辑设备名:逻辑设备名称是首次将设备添加到系统时创建的。在大多数文件系统命令中,可以使用逻辑设备名称引用相应的设备。
3. 磁盘低级格式化:将硬盘划分出柱面和磁道,再将磁道划分为扇区,以及对扇区的进一步划分
4. 磁盘高级格式化:清楚硬盘上的数据,生成引导信息,初始化FAT表.
5. 柱面斜进量:访问相邻的柱面时,需要花费一定的时间从一个柱面移动到另一个柱面,这期间花费的时间磁盘转过的距离所导致错开的首扇区位置.
6. 地址映射:控制器实现虚拟几何规格到物理地址的映射.
7. 盘块(簇):一组相邻的扇区,是内核I/O子系统的磁盘传输基本单位.
8. RAID:廉价磁盘冗余阵列(Redundant Array Inexpensive Disk),将一份数据分散到不同的磁盘,实现并行存取,提高速度和可靠性.
9. 驱动程序:直接控制设备的程序称为驱动程序,可以直接读写控制器中的寄存器.一般运行在内核态.
10. 内核I/O子系统:进行有关设备共性的操作,在输入输出层次模型中位于用户及I/O软件和驱动程序之间,完成缓冲区管理,缓存管理,I/O保护和错误处理等功能.
11. 设备相关:程序只能在特定的设备环境下运行,称为设备相关的;否则称之为设备无关的.
12. 阻塞I/O(blocking IO):请求I/O系统调用,一般情况下进入等待,即被阻塞.当设备返回结果时,进程就变为就绪状态.不等待的情况是,数据已经在缓存中了.
13. 非阻塞式I/O:执行I/O操作时立即返回,操作的结果对于后续的操作可有可无.对于单线程,有多少读多少,对于多线程,立即返回执行其他线程.
14. 异步(synchronized)I/O:进行I/O操作,立即返回,进程继续执行.当I/O操作完成后,向进程发送信号,执行回调函数.
15. I/O流:将I/O过程分为若干个阶段,每个阶段由一个模块负责,模块能被不同的输入输出过程共享.

输入输出

1. 外设与CPU的三种链接方式? 答:总线,控制器,端口
2. 设备的控制方式? 答:轮询(短时),中断(开销较大),DMA(块设备).
3. 设备地址由ISA所规定.两种编址方式?
答:统一编址,独立编址. 各自的优缺点?
4. 向磁盘的某个位置读写数据的原则:
同一个盘面同一个磁道优先,然后是同一个柱面,再然后是不同柱面.
5. 给定三维的地址时,查找顺序: 柱面->磁头->扇区
6. 从块号,到(柱面/磁道/扇区)的映射,是由**驱动程序**实现的.
7. 映射公式:

1 | 线性地址<==>(柱面号×每柱面磁头数+磁头号)×每磁道扇区数+扇区号

磁盘调度

1. 为什么要进行磁盘调度?

当有一组请求时,可以选择服务的先后顺序,从而更加高效地完成任务.
(主要控制磁头移动的顺序)

2. 常用的磁盘调度算法?

- FCFS(符合初始认知,无饥饿现象)
- 最短寻道时间优先(SSTF shortest search time first):效率提高,但是存在饥饿
- SCAN(电梯调度算法):沿着一个方向到最后一个柱面!,再折返.**会产生饥饿现象吗?**
不仅考虑到欲访问的磁道与当前磁道间的距离,更优先考虑了磁头当前的移动方向;避免了出现“饥饿”现象。
- CSCAN(电梯调度地改进):返回途中不响应任何其他请求.
- LOOK:到最后一个请求为止
- CLOOK:到最后一个请求,并且返回途中不响应任何其他请求

时钟+驱动

1. 时钟的基本功能

- 获取当前时间
- 计时(计算时间片)
- 在时刻t触发某些操作

2. 驱动程序与设备关系

根据主设备号找到驱动程序,将次设备号作为传入传递给驱动程序.

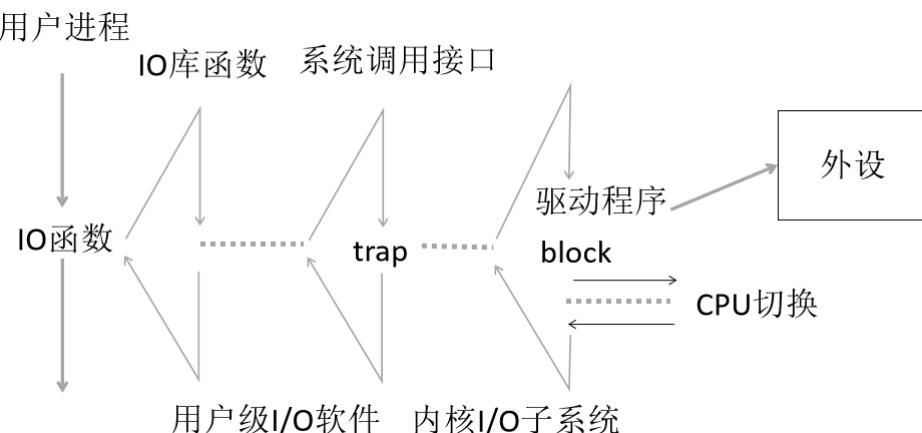
3. 驱动程序是设备相关的(设备依赖的).直接控制设备,与设备特性紧密相关.

4. 驱动程序的结构和功能:

启动设备(启动部分):驱动程序被内核I/O子系统调用

中断服务:可以被中断处理程序调用,也处理来自于设备的中断事件.

初始化:确定中断号,注册服务例程(挂在中断号对应的服务例程链表中).



5. 驱动程序与设备:

发送命令,检测状态,通过中断机制响应设备事件

6. 中断处理程序与中断服务例程:

中断处理程序对应一个中断号,一个中断号对应一条中断线,一条中断线上可能存在多个中断服务例程.也就是说,一个中断处理程序可能对应多个中断服务例程.

7. 内核I/O子系统是设备独立的,但是与驱动程序有接口.

8. 为什么中断服务例程的上半部分完成后要开中断?

如果关中断的事件太长,CPU就不能及时响应其他的中断请求,从而造成中断的丢失,因此内核的目标是尽可能快的处理完中断请求,尽其所能将更多的处理向后推迟.

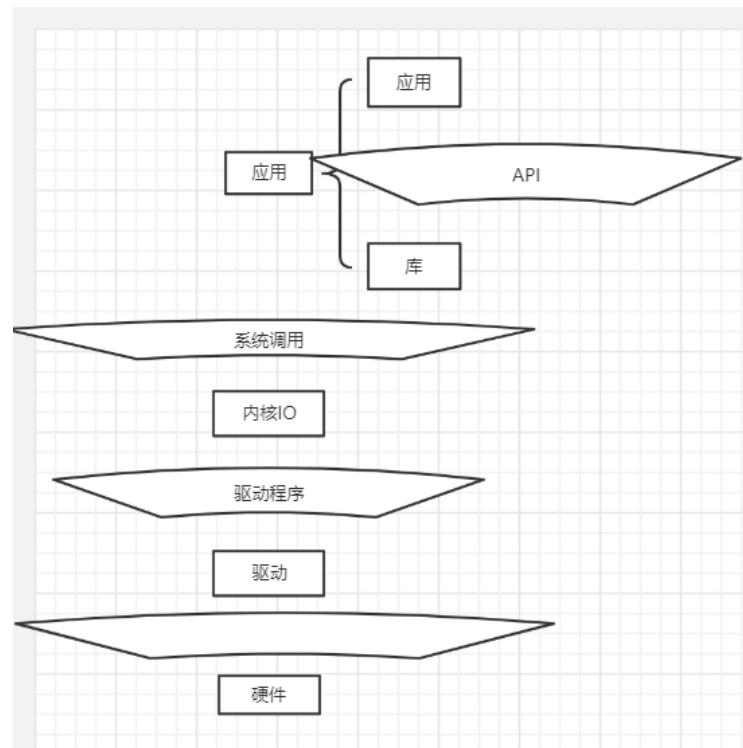
9. 高速缓存与缓冲区的区别?

作用:高速缓存为了提高效率,而缓冲区用于通信双方的数据交换.

内容:高速缓存的内容是另一个存储器内容的备份.缓冲区内的内容是唯一的.

buffer-cache:例如内存中的文件块.

10. IO软件接口类型



- 驱动程序:驱动程序与内核IO子系统的接口,封装设备特性
- 系统调用:封装内核IO子系统的实现
- 应用接口API:为应用程序提供跨平台IO服务.例如一份代码在多设备皆可用.

习题解析

1. 试说明I/O系统的基本功能。

- 隐藏物理设备的细节
- 与设备的无关性
- 提高处理机和I/O设备的利用率
- 对I/O设备进行控制
- 确保对设备的正确共享
- 错误处理

2. 简要说明I/O软件的4个层次的基本功能。

- 中断处理程序: 用于保存被中断进程的CPU环境, 转入相应的中断处理程序进行处理, 处理完后恢复现场, 并返回到被中断的进程
- 设备驱动程序: 与硬件直接有关, 用来具体实现系统对设备发出的操作指令, 驱动I/O设备工作
- 设备独立性软件: 用于实现用户程序与设备驱动器的统一接口、设备命令、设备保护, 以及设备分配与释放等。
- 用户层I/O软件: 用于实现用户与I/O设备交互

3. I/O系统接口与软件/硬件 (RW/HW) 接口分别是什么接口?

- I/O系统接口是I/O系统与上层系统之间的接口, 向上层提供对设备进行操作的抽象I/O命令, 以方便高层对设备的使用;
- 软件/硬件 (RW/HW) 接口的上面是中断处理程序何用于不同设备的设备驱动程序, 它的下面是各种设备的控制器。

4. 与设备无关性的基本含义是什么? 为什么要设置该层?

- 为了提高OS的可适应性和可扩展性, 在现代OS中都毫无例外地实现了设备独立性, 也称设备无关性。
- 基本含义: 应用程序独立于具体使用的物理设备。为了实现设备独立性而引入了逻辑设备和物理设备两概念。在应用程序中, 使用逻辑设备名称来请求使用某类设备; 而系统在实际执行时, 还必须使用物理设备名称。

- 优点

- 设备分配时的灵活性
- 易于实现I/O重定向（用于I/O操作的设备可以更换（即重定向），而不必改变应用程序。）

5. 试说明设备控制器的组成。

答：设置控制器与处理机的接口；设备控制器与设备的接口；I/O逻辑。

6. 为了实现CPU与设备控制器之间的通信，设备控制器应该具备哪些功能？

基本功能：接收和识别命令；数据交换；标识和报告设备的状态；

地址识别：数据缓冲；差错控制。

7. 什么是内存映像I/O？如何实现的？

通过将外围设备映射到内存空间，便于CPU的访问（即统一编址方式）

8. 为什么说中断是OS赖以生存的基础？

中断在操作系统中有着特殊重要的地位，它是多道程序得以实现的基础，没有中断，就不可能实现多道程序，因为进程之间的切换是通过中断来完成的。

另一方面，中断也是设备管理的基础，为了提高处理机的利用率和实现CPU和I/O设备并执行，也必需有中断的支持。中断处理程序是I/O系统中最低的一层，它是整个I/O系统中最低的一层。

9. 对中断源的两种处理方式分别用于那种场合？

- 屏蔽中断：当处理机正在处理一个中断时，将屏蔽掉所有的中断，直到处理机已处理完本次中断，再去检查是否有中断产生。所有中断按顺序处理，优点是简单，但不能用于实时性要求较高的中断请求。
- 嵌套中断：在设置了中断优先级的系统中，当同时有多个不同优先级的中断请求，CPU优先响应优先级最高的中断请求，高优先级的中断请求可以抢占正在运行的低优先级中断的处理机。

10. 设备中断处理程序通常需完成哪些工作？

- 1、唤醒被阻塞的驱动进程。
- 2、保护被中断进程的CPU环境。
- 3、转入相应的设备处理程序。
- 4、中断处理。
- 5、恢复被中断进程的现场。

作业题

1.

- (1) 页调度和置换在原则上就应该优先于用户的请求，否则无妨响应用户请求。
- (2) 开机时内核的初始化时所需要的IO原则上高于用户IO。
- (3) 某些紧急事件，比如说某些不及时处理就会被冲刷的IO，此时OS会优先处理这些紧急事件而延后用 户IO。