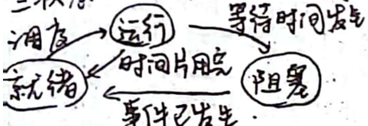
程序② 一个正在计算机中运行的程序实例③ 能分配给处理器并由处理器执行的一个实体④ 由一个顺序执行的代码组、当前状态和相关系统资源刻画的活动单元.

为什么引入进程？"作业""进程"等概念不能有效描述程序动态轮换执行的过程. 资源拥有者的身份

进程5大特征 ① 动态性. 因创建而产生，因调度而执行，因得不到资源而暂停，因完成而消亡 ② 并发性. 并发执行 ③ 独立性 进程是程序运行的基本单位. 也是资源分配和调度的单位 ④ 异步性. 进程速度未知. ⑤ 交互性. 相互制作用

三状态



状态

建：建立PCB，但进程其他内容未入主存 退出：进程终止，资源等回收

建进程原因：① 新批处理作业 ② 交互登录 ③ 由现有进程派生

止原因：正常完成/超时限制/阻碍

状态

行→就绪挂起：直接换出

挂起进程特点：① 不能立即执行 ② 能在/不在等待一个事件，若等待，则阻塞条件不依赖挂起条件，阻塞事件发生不会使进程立即执行 ③ 为阻止程序运行，可通过代理置于挂起 ④ 除了代理显式命令，否则进程无法转移.

7. 挂起原因 ① 交换：系统需释放足够内存空间 ② 其他OS原因 进

请求 ④ 定时：周期性执行 ⑤ 父进程请求：编程的需要

8. 进程映像内容 ① 一段可执行代码 ② 程序所需数据 ③ 上下文环境和运行时信息. 表征进程 ① 标识符 ② 状态 ③ 优先级 ④ 程序计数器 ⑤ 内存指针.

9. 进程是资源的拥有者. 信息表内容：① 内存表 ② I/O表 ③ 文件表 ④ 进程表

10. PCB：进程存在的唯一标志，可访问进程所有信息，记录上下文环境

11. PCB内容：① 进程标识信息 ② 处理器状态信息 ③ 进程控制信息. ① 进程标识符 ② 一用户可见寄存器、状态寄存器、栈指针 ③ 一状态信息、进程间通信

12. 内核模式 — 调用一个操作系统的服务或中断来触发系统例程的执行时 用户模式 — 从系统服务返回到用户进程，置为用户模式

13. 进程创建流程：① 为新进程分配唯一标识符 ② 为进程分配空间 ③ 由父进程参数将子进程PCB初始化 ④ 将此PCB插入就绪队列 ⑤ 创建扩充其他数据结构

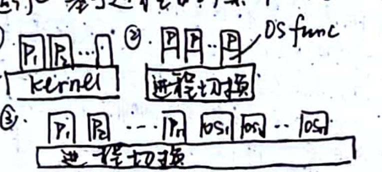
14. 进程切换：① 保存处理器上下文 ② 更新当前进程的PCB ③ 把进程PCB移到相应队列 ④ 选择另一进程 ⑤ 更新内存管理数据结构 ⑥ 恢复现场

15. 进程撤销. 2种策略：只撤销具有指定标识符 / 撤销指定进程及子孙进程. ① 找到被撤销进程PCB ② 立即停止进程执行 ③ 有子孙进程？撤销. 对后一种策略) ④ 所占资源归还给父进程/系统 ⑤ 撤销PCB

16. 阻塞：① 由于进程处于执行状态，故先中断处理机，保存现场 ② 将该进程插入等待队列中 ③ 就绪队列中选一个进程投入运行 | 由运行→阻塞状态，是进程自己调用阻塞原语

17. 唤醒：阻塞→就绪，由发现者进程调用唤醒原语 与当前进程合作

18. 操作系统与进程关系 ① 无进程内核 ② 操作系统例程在用户进程内运行 ③ 基于进程的操作系统



19. 进程创建 — fork() 撤销 — exit 阻塞 — sleep, wait 唤醒 — wakeup

20. 线程引入：① 进程的核心任务：争夺CPU ② 进程通信本质：循环调用

进程的特点 ①. 资源所有权 存放进程映像的虚拟地址空间，资源控制权，预防资源冲突 ② 调度：不同进程的执行过程交替执行. 进程是调度的实体

27. 线程引入：保留并发的优点，避免了进程切换代价. 实质是内存映射表与资源不变，而PC和寄存器变

28. 进程是资源分配、保护单元：容纳地址空间(容纳进程映像). 对处理器的受保护访问(和I/O资源)

29. 每个线程有 ① 执行状态 ② 线程上下文 ③ 用于局部变量，静态存储空间 ④ 与进程其他线程共享内存、资源访问

30. 多线程优点 ① 响应度高 ② 资源共享 ③ 经济快捷 ④ 方便多CPU处理

31. 线程状态：① 运行 ② 就绪 ③ 阻塞 操作：① 派生 ② 阻塞 ③ 解除~ ④ 结束

32. 线程分类 ① 用户级 ② 内核级~ 2种线程都是用户创建. 区别：① 内核：操作系统创建对应数据结构直接管理，以线程为调度单位 ② 用户：不感知线程，以进程为单位

33. 用户级线程：① 管理线程的工作都由应用程序完成. 内核意识不到线程的存在. 所有活动发生在用户空间 线程库：创建销毁进程，调度线程执行

34. 用户级优点：不需切换为内核模式，可为程序定做调度算法 可在任何操作系统中运行. 缺点：系统调用阻塞进程中所有线程，不能利用多处理器

35. 内核级线程：33 ① 由内核完成 ② 内核为进程、线程维护上下文信息 ③ 调度由内核基于线程完成 ④ 线程切换需到内核模式

36. 进程同步：多个进程在执行次序上进行协调，使并发的诸进程能合作，使程序执行具有可再生性. 为什么同步？① 多个进/线同时完成1个任务，② 除了并行工作外，还出现相互等待的协作过程

37. 竞争条件：多个进程/线并发访问同一数据. 结果与执行顺序有关.

38. 临界区. 把一次仅允许一个进程用的资源→临界~. 访问临界~→临界区

39. 互斥. 一个进程在临界区访问共享资源时，其他进程不能进入该临界区访问共享资源. 3条件：① 互斥进入 ② 有空让进 ③ 有限等待

39. 算法1：① 不满足有空让进 ② 若在临界区结束，则另一进程饿死 算法2：不能互斥访问 算法3：

40. 死锁：2个进程以上/进程等待其他进程完成某些事情，而不能继续执行的情形. 活锁：2个进程响应其他进程的变化而改变状态，但不

41. Dekker P0： 有用2个

```
flag[0]=true; while(flag[1]){
if(turn==1){flag[0]=false; while
(turn==1); flag[0]=true;}}
临界区 turn=1; flag[0]=false
其余部分  P1代码：0->1, 1->0
Peterson P0: flag[0]=true; turn=
while(flag[1]&& turn==0); 临界区
flag[0]=false; 剩余部分 P1: 0->
                            1->
面包店 choose[i]=true; num[i]=
(Pi) max(num[0]..num[n-1])+
choose[i]=false; for(j=0 to n-1)
while(choose[j]); while((num[j]
&&(num[j],j)<(num[i],i));}
临界区 num[i]=0; 剩余部分
```

42. 中断禁用. 通过阻止其他进程执行的方法，来达到阻止进入临界区 cli 关中断 sti 开中断 对多处

43. 专用机器指令. 比较和交换指令/exchange 比较和交换指令：c(*w,t,n). 若*w=t, 则*w=n 返回*w原来的值 优点：适用多处理器；简单；多个临界区 缺点：忙等待，饥饿，死锁

44. 饥饿：由于别的并发激活进程持久占资源，某个异步进程在不可预知的时间内不能被激活的情

45. 原子操作：不会被线程调度打断的操作，一旦开始不会切

46. 信号量. ① semWait −1，变负数则阻塞执行进程，否则继 ② semSignal +1，值≤0，则被semWait阻塞的进程解除阻塞 信号量初值 S=1?→只有1个进程

47. 二元信号量. 一样. S=0/1 sem 1→0 0→阻塞 semSignal：若阻 唤醒. 否则0→1

48. 强信号量. 进程从阻塞队列出遵循FIFO，可避免发生饥

49. 有限缓冲区生产、消费者问题

```
s=1, n=0, e=buffersize(sem
producer: while(true){prod
semWait(e); semWait(s); app
semSignal(s); semSignal(n);
consumer: while(1){sW(n);
take(); sS(s); sS(e); consume
```

无限？去掉e即可 二元信号量

append之后：no++; if(no==1) sS(

take之后：no--; m=no; 在最后：if(!n

50. sW, sS的实现 sS(n) sW

compare&swap：以sW为例

开头加一句 while(c_&_s(s.fl 0,1)==1); 结尾加句 s.flag=0

send, ~receive；无~send, ~receive；
无~send, 无~receive

2 使用消息的互斥：message msg；while(1){receive(box,msg)；临界区 send(box,msg)；其余部分}}
main(){create_mailbox(box)；send(box,null)；

3. 消息，有界缓冲：生产者/消费者：
producer(){message pmsg；while(1){receive(mp,pmsg)；pmsg=produce()；send(mc,pmsg)；}}
consumer(){message cmsg；while(1){receive(mc,cmsg)；consume(cmsg)；send(mp,null)；}}

读/写者问题：任意数量的读进程可以同时读，1次只有1个写进程可写，若1个写进程修改，则禁止读进程读取。读优先：
reader(){while(1){sW(x)；rc++；if(rc==1) sW(wsem)；sS(x)；READ()；sW(x)；rc--；if(rc==0) sS(wsem)；sS(x)；}} writer(){while(1){sW(wsem)；WRITE()；sS(wsem)；}}

固定分区：区大小可以相同/不同。存在问题：程序太大无法装载到同一区：部分装载+覆盖；内存碎片。放置算法：放到容纳的最小分区中。

动态分区：分区长度和数量可变，外存碎片，需压缩。放置算法：最佳适配/首次~(从开头)/下次~(从当前)。伙伴系统：①可用内存块大小 2^K，$L \le K \le U$ ②每个大小为 2^i 的内存记录在一个列表中 ③请求大小为s块，确定i，有 $2^{i-1} < s \le 2^i$ ④用递归的放置算法，找到内存 ⑤合并空闲伙伴，移入空闲列表

伙伴系统二叉树：叶子结点代表……，深度代表块大小 175

逻辑地址：每个用户进程看到的独立地址空间。相对地址：相对于进程首部分(起始地址)的地址。线性地址：逻~、线~变换的中间层，分页系统才存在线性地址

分页：进程的块：页，内存的块：页框，1个进程可以占多个不连续页框，最后一页有内碎片，无外碎片

页表：每个进程每页对应页框的位置。页、页框大小为2的幂；逻辑地址→物理地址的映射：①页号，查页表得到物理页号 ②加页偏移→页地址

2. Linux0.11中，页框号要去掉后12位！
3. 分页对程序员透明，分段可见
4. 分段：需段表，由段号得段表项：长度&基址，偏移量<长度，再加上基址→物理地址

……进程中所有内存访问都是逻辑地址，运行时动态转换为物理地址

65 局部性原理 ①某条指令执行后不久可能被再次执行 ②一段时间内访问的地址往往集中在一定范围

66 调页时机：请求调页/预调页 [illegible] 在内存的逻辑地址→缺页中断 [illegible] 当块被装载入内存→I/O时间长，调度其他进程

67 驻留集：进程被分配的页框的集合

68 内存分配：固定/可变分配(页框数是否固定)，内存置换：全局/局部置换(是从所有/被分配页框中选择被置换的页)

69 虚存的效果：①支持更有效的多道 ②进程可以比内存全部空间大

70. 虚存手段：①段页管理 ②部分加载 ③按需调页 ④换入换出

71. 系统抖动：内存空间占满时，系统和一块时必须换出另一块，若又用那块，换很快取回。大部分时间用于交换块而不是执行。解决方法：局部性原理

72. 虚存中的地址：

页号	偏移

页表项(PTE)

修改	存在	其他控制	框号

页号→PTE→框号 + 偏移→物理地址

73. 页表很大？虚存/多级页表

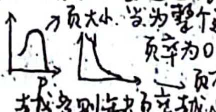
74. 2级页表：先由页号1→PTE1→页表2的基址，再由基址+页号2→框号

75. 倒排页表：页号→hash→某条PTE→链接到另一条PTE→框号 181

76. TLB：页表的cache

77. 页表项可在TLB、内存、磁盘中，访问的数据可在cache、内存、磁盘

78. 页尺寸：越小→内碎片少，页表越大，可用内存少，可能导致2次缺页
越大→缺页中断有增加的可能(数据越集中)，接近进程大小，缺页率为0



当为整个进程大小时，缺页率为0

页框分配的数量，越多则缺页率越小

79. 分段系统：无明显差异。段号→段表→长度，基地址→基地址+偏移

80. 段页式：段号因与页的基地址→页号→页表→页框号+偏移→物理地址

81. 置换算法 ①LRU：上次未使用最早 ②FIFO ③CLOCK：换入时记为1，替换为时开始扫，1→0，0→替换，并再次移动一下指针

82. 请求调页的外存：①文件区(存放不被修改的文件) ②存放交换页面的交换区

83. 页面置换：①足够多交换区：从交换区调入所需页面，将进程文件复制到交换区 ②缺少交换区：不会修改的文件：直接从文件区调入，可能修改的：换出时放到交换区，需要时调入 ③UNIX方式：未运行过页面：从文件区调入，曾运行过但被换出的页面→交换区，下次调入时从交换区调

[illegible] 驻留集：[illegible]进程分配的物理内存空间 ①分配给1个进程的物理内存越小，驻留在内存中的进程越多，提高处理器利用率 ②页框过少(单进程)，缺页率高 ③超过某个值，由局部性原理，对进程缺页率没明显影响 [illegible]

84 长程调度：决定进入内存的作业，选择一个或一批，建立进程加入[illegible]，频率低，用于控制并发度。中程调度：进程从内存换出，重新调入内存并从中断处执行，发生于不使用虚存的系统。短程调度：将处理机分配给一个处于就绪态的进程，分为抢占/非抢占。能有效提高处理机利用率和响应效率，通常意义的调度

85 FCFS：对长作业有利，短~无利

86 短作业优先(SJF/SPN)：选择时间最短的作业调入内存，平均等待时间、周转时间最少，不利于长作业

87 最短剩余时间(SRT)：抢占+SPN = 最早截止时间(剩余时)

88 高响应比(HRRN)优先：响应比 = $\frac{\text{等待时间}+s}{s}$，s：要求服务时间

89. 优先级调度：每次从就绪队列中选择最高优先级的进程 183

90 RR：适用于分时系统，程序选择队列中第一个进程执行，仅运行1个时间片

91. 反馈：多个就绪队列，每个队列优先级不同，优先级越高时间片越小，在队列内FCFS。一个进程被抢占→转入下一级。优先级高队列为空→调度下一级队列，最后一级？放最后一级

92. 轮转/SRT/反馈：会抢占

93. 对称多核调度器：主流处理器，每个CPU自己决定调度算法。非~：仅1个处理器能处理系统数据结构。软/硬亲和性：进程尽量不/不切换

---

94. I/O控制：设备驱动和中断处理程序，在内存、磁盘之间传送信息。基本I/O管理：将逻辑块地址转化成物理块地址，空闲空间管理。逻辑I/O：管理文件系统的逻辑结构信息，负责文件保护

95. 文件组织：①堆：按顺序到达收集，每条记录长度可变 ②顺序文件：每条记录有固定格式，长度相同，包含相同数量和长度相同的域 ③索引顺序文件：用索引提供快速接近目标记录的查找能力；增加溢出文件，用于记录新增、删除、修改记录 ④索引文件：对可变长记录文件，建立表项大小相等的索引表，可以多索引

96. FCB：存放控制文件所需信息的数据结构，如文件名、位置、存取权限、建立时间。目录也是文件(目录文件)

97 [illegible]

索引节点(FCB) [illegible] 每个目录项都是目录文件[illegible] 索引节点：除文件名之外的其他描述信息单独形成1个数据结构，为索引节点 [illegible] 目录文件的目录项仅由文件名、指向索引节点的指针构成，每个文件在磁盘中有唯一的索引节点，文件被打开时，存入内存

100 文件系统在内存的结构 ①打开文件表：文件名/读写指针/访问权限/系统打开索引号 ②系统打开文件表(系统只有1张)，有外存地址/打开计数

101 文件系统在外存的结构
MBR—引导区—超级块—[illegible]
MBR：位于0号扇区，记录分区表和有关信息，分区大小及位置信息。引导块：引导块中程序负责启动该分区OS。超级块：包含文件系统的关键信息，启动时读入内存，是文件系统第1个块，存放文件系统结构信息，系统整体信息。空闲块：用位图/指针链接

102. 文件共享机制 ①基于i节点的文件共享(硬链接) ②利用符号链接实现~

103. 逻辑组块：在外存管理中，文件的逻辑地址空间被分为一个个文件块

104. 组块方式：定长组块，变长跨越、变长非跨越。变长跨越式组块存储效率高，实现难度大。非跨越：浪费

105. 辅助管理：为文件分配磁盘块，辅存空闲块的组织，分配、回收

106. 连续文件分配：速度快，直接访问，长度不宜增加，外碎片

107 隐式链接：无外碎片，只适合顺序访问，稳定性差

108. 显式链接：将链接各物理块指针显式存在1张表中，即FAT；整个磁盘仅1张FAT，FAT可标记空闲磁盘块，会占用较大磁盘

109. 索引分配：索引表中记录了文件的逻辑块对应的物理块，索引表放在索引块中，每个文件必须有一个索引块

110. 混合索引分配：UNIX采用，有直接地址/1次/2次间接寻址

111. 空闲块管理 ①空闲位图 0→占用 1→ ②空闲链表 ③FAT ①快速连续分配，位向量大 ②分1个快，分多个慢

112 MINIX系统：引导块+超级块+i节点位图+逻辑块位图+i节点+数据
超级块：i节点数/逻辑块/位图所占
逻辑块位图：标记磁盘分区中每个数据块使用情况，只标记数据区盘块
i节点：存放文件系统中目录文件和普通文件的i节点，第1个i节点：根目录，内容如：文件类型属性，直接，一/二次间接块

114. 如何访问/usr/bin/文件名3？
①根目录文件的i节点固定，即第1号i节点 ②读i节点的数据块内容为根目录下的目录项编号 ③通过该目录项而获得/usr ④若找到，则用

节点的数据块，可以获得/usr目录文件的内容，即子目录usr的文件目录项列表 ⑥通过…匹配"bin" ⑦若…/usr/bin…i2 ⑧根据i2 …/usr/bin内容…⑨通过…匹配"文件名" ⑩若…/usr/bin/文件名的…i3 ⑪根据i3…可取得文件"文件名"的内容

5. 如何删除文件 /usr/bin/vi
①-⑪同前 ⑫根据文件长度从i3节点中找到各个盘块的索引号，将i节点位图和逻辑块位图中相应位设置为0（释放）⑬删除目录文件bin中对应表项

linux 0.11中的文件建立：
①将用户进程打开的文件在filp中进行登记 ②建立filp与filetable进行连接 ③将要打开文件的i节点在filetable中登记 ④将新打开的i节点登记到inodetable

7. I/O设备分类：①人可读 ②机器可读 ③通信设备(网卡) 或：①块设备 ②字符设备 或：①低速(键盘) ②中速(打印机) ③高速(磁盘).

181

I/O控制方式：①程序直接控制 设备控制器，非中断程序 ③通道控制器，中断 ④DMA直接到存储器 I/O通道：I/O模块有单独处理器，有专门为I/O指令设计的指令集，CPU指导I/O程序完成一系列I/O活动 I/O处理机：模块有局部存储，像一台计算机

的I/O的逻辑结构
户进程—逻辑I/O—设备I/O(设备独立性软件)—中断处理
逻辑I/O：把设备当成逻辑资源处理. 设备I/O：请求的操作和数据转换成适当I/O指令 度与控制：I/O操作的排队调度、中断. 是I/O模块与硬件真正发生交互的软件层
其他层：目录管理，文件系统(处理指定操作)，物理组织(逻辑地址→物理地址)

2. I/O缓冲：提高CPU与I/O的并发性，解决二者速度不匹配.
缓冲：I/O设备 → □ 单缓冲，操作系统 → 用户进程.
双缓冲：2个缓冲区. 外设和CPU交替使用这2个缓冲区
循环缓冲：内存中多个大小相等的缓冲区，将其链接为环形链表

123. 缓冲池 ①当输入设备需数据输入时，则从空缓冲队列的队首取下一个空缓冲区，作为收容输入工作缓冲区，被装满数据时，则将该缓冲区挂到输入缓冲队尾 ②当某进程从缓冲池读入数据时，则从输入缓冲队列的队首取一个缓冲区作为提取输入缓冲区，取完后，将此缓冲区链接到空缓冲队列队尾 ③同①，输入→输出，收容输出工作缓冲区，该进程向该缓冲区中存放数据，装满数据后则链接到缓冲区队尾 ④当输出设备需进行数据输出时，则从输出缓冲队列的队列的队首取一个缓冲区作为提取输出工作缓冲区，并从中提取数据输出，取完后，则将被缓冲区链接到空缓冲区队尾

124. T磁盘访问 = 寻道 + 旋转延迟 + 传输时间
寻道时间 $T = m \times n + s$ → 启动磁臂时间；$m \times n$：跨越n条磁道时间
旋转延迟 $T_r = \frac{1}{2r}$. r—转速
注意单位. 若r为rpm → $T_r = \frac{60}{2r}$
传输时间：$T_t = b/rN$ b：读写字节数 N：磁道上字节

125. 磁盘调度：
FIFO：按时间顺序调度
SSTF：每次从当前位置选择移动时间最少的IO请求
电梯 SCAN：磁头仅沿1方向移动，直到最后1个磁道，再反转方向
循环电梯 C-SCAN：把扫描限制在1个方向上，扫描完其他请求时返回相反方向末端

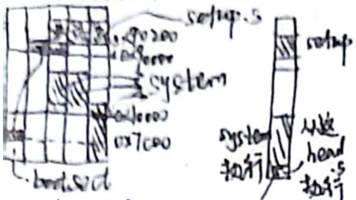
126. 磁盘缓存：利用内存中的存储空间来暂存从磁盘读出盘块的信息，逻辑上属于磁盘，物理上驻留内存
形式：①内存中开辟一个单独空间 ②把未利用的内存空间作为缓冲池.

127. 对哪些实体，有信息表？内存，I/O，文件，进程 128. 为什么有2种模式(用户/内核)：用户模式下指令，访问内存区域都受限，为了防止OS受到破坏. 内核模式无限制
129. 三个中断例：时钟中断，I/O中断，内存失效
130. 中断、陷阱区别：中断为与当前进程无关的某些类型的外部事件相关. 陷阱与当前进程的错误相关
169
131. 模式切换与进程切换区别？模：不改变当前运行状态进程的状态. 进：一个正在执行的进程被中断，OS指定另一进程为运行态. 进程切换保存更多状态信息
132. 就绪态不能转换为阻塞态：处于就~进程无法执行任何造成阻塞代码(系统调用)，故不行. 阻塞态不能转换为运行态？因为被调度的进程才会转入运行态，只有就绪态才会被调度. 因此阻塞态必须经就绪态才能运行态

133. 为什么定义进程状态 进程执行的间断性，决定了进程可能具有多种状态 ①就绪态—某时刻进程已获得处理机之外的所有资源，一旦分到处理机就可执行 ②运行态—进程已获得必要资源，占有处理机运行. 阻塞—正在执行的进程，因发生某事件暂时无法执行
134. 线程例子：①前台后台操作 ②异步处理 ③加速执行 ④模块化结构
135. 被进程所有线程共享：地址空间、文件资源、执行特权 开销低 178
136. 用户进程的特点：①进程不需要为了线程管理切换到内核模式，节省了2种模式的切换 ②调用可以是程序专用的 ③用户级线程不用对底层内核修改，可在任何OS运行
137. 用户级线程缺点：①用户级线程执行系统调用时，不仅此线程阻塞，进程中所有线程都会阻塞 ②一个进程中只能有1个线程可以执行.
138. 为什么可用关中断(cli)和开~(sti)来实现临界区？局限性？由于临界区不能被中断，故可实现互斥. 不能用于多处理器体系结构中
139. 为什么信号量的操作必须为原子操作？①若信号量操作可中断，则多个线程试图同时访问或修改同一资源，可能发生竞态条件 ②若不是原子，无法正确保证进程间的正确同步与互斥，有竞争死锁
140. 管程：一种程序设计结构
特点：①局部变量只能被管程的过程访问 ②进程通过调用管程的一个过程来进入管程，任何时候只能有1个进程在管程执行，其余阻塞
141. 管程优点：所有同步进程限制在管程内部，易于验证同步，检验错误
142. 虚存定义：对内存采用分页/分段管理后，①进程中所有内存访问都是逻辑地址，运行时动态转为物理地址，使重定位成为可能 ②进程被划分成许多块，不需连续存
143. 请求调页：当访问某页的内存中一个单元时才将该页调入内存，仅在需要时加载页
144. 预调页：读取的页不是缺页中断请求的页
145. 处理器访问不在内存的逻辑地址：①产生缺页中断 ②若内存已满，先置换 ③进程被阻塞，OS发出磁盘I/O请求 ④ [illegible]

内存后，修改页表，通过I/O阿 唤醒该进程 [illegible]
146. 举例说明：多级页表相对一级页表的优点：假设页大小4K，一个的虚拟地址空间为4GB，进程用4MB内存空间. 对于一级表，4M空间存放4GB虚拟地址对应的页表(1页表项4B，1页为1K个页表项，需4M/1M=4个页表项，大小4MB)
二级目录：1个页目录项相当于4M空间，存放1个页目录项占4K(4K/4个表项，4M×1K=4G) 还需要1页于存放进程使用的4M空间对应页表. 共需4K+4K=8K来存放
147. 系统调用：吞吐量 = $\frac{总作业数}{总时间}$
CPU利用率 = $\frac{忙碌时间}{总时间}$
148. 页大小为4K，物理内存中有进程，且实际物理内存的未分空间为14K，现有1个编译好的程序test放在指定路径…简
①文件执行，有文件的全路径所以一级一级打开目录文件，直到它的i节点
②根据i节点中的混合索引确定文件长度和各盘块的位
③创建进程控制块，分配进程
④空间不够，题目未说虚存，以中程调度，将其它进程换
⑤按段页式，分配内存空间，
⑥启动I/O，读入文件内容 地址映
⑦CPU调度，计时器中断，进
⑧执行结束，系统调用，设 变进程状态为退出，释放内存 如需要，I/O请求，写盘
149. I/O调度的优缺点：①FIFO 进时性能好，多时接近随机调度 ②SSTF：从当前位置移动最存在饥饿 ③SCAN：对最近跨的区域不公平，不能很好利用局
150. 提升寻道时间：使用更小更轻的件，如降低磁盘直径，减少磁头
151. 提升旋转延迟：r↑ 距
152. Linux 0.11 内核结构：
①系统调用是用户与系统间的唯一接
②虚拟内存管理，一切皆文件，高速缓存 153. 程序结构：
bootsect → setup → syste
其中system中含有head和main
bootsect+setup+sys=内核映像
153. system中全部内容：head、main Kernel、mm、fs、lib
154. 执行顺序：ROM BIOS
bootsect.s → setup.s → head.s → ma
(system模块) in.c

0x7c00 → 0x90000，跳去setup
执行：system模块从0x10000 → 0x0
加载模块，读取计算机数据



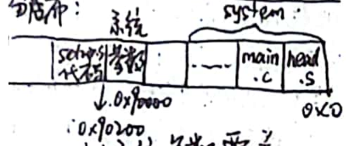
引导启动程序—包含bootsect、setup

为什么bootsect把自己移到了0x90000？system放置在0x10000处，假设system模块最大长度不超过0x80000（512KB），所以从0x10000~0x8ffff都预留给system模块，不会到0x90000，所以bootsect会移到90000，把setup

为什么load system不放在0x0？加到后面

开始0x0放bios中断向量表，现在bios不用了，可以覆盖了。

6.引导启动之后，内存分布：



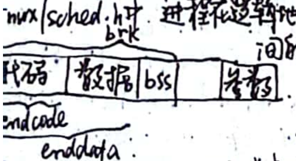
bootsect被系统参数覆盖

7. head.s的内容：设置段寄存器、初始化idt、gdt、设置管理内存的分处理机制，将页表放在main.c程序入口地址压栈，跳转过去执行

main.c内容：①根文件系统是INIX ②文件：/etc/rc、/bin/sh、/dev/* 目录：/etc/、/dev/、/bin/、/home/、/home/root/.

main.c流程：系统初始化→物理存各部分进行功能划分→系统各部分始化→移到任务0执行→创建进程→空闲时执行pause()→加载根文件统→设置终端标准I/O→创建程2→循环等待2退出→创建子程→循环等待进程退出

进程控制块.task_struct.位于nux/sched.h中 进程在逻辑地址空间的分布.



1.信号提供了1种处理异步事件的方法.处理：①忽略②捕获③执行默认操作

2.系统调用：system_call，nt 0x80，时钟中断，0x20

3.程序中的几个段：①-⑤五段
①bss：未初始化全局变量、静态内存分
②data：已初始化······→ 配
③text：执行代码的一块内存区域，也可能有常数变量（字符串常量）
④堆：动态分配，malloc.
⑤栈：临时创建局部变量.
一个程序本质由bss、data、text

同一线程中，另一个用户级线程进行时则线程被阻塞，整个进程都被阻塞（多对一，只有1个内核级线程，OS内核只能看到1个调度单位）线程被阻塞则整个进程被阻塞

165. FCFS适用CPU繁忙型，不利于I/O繁忙型

166. 不会产生内部碎片：动态分区、分段

167. 无动态重定位：固定分区

168. 页面大小对分页sys影响？页面大，则内存碎片大，小则页表长

169. 影响请求分页系统平均访存时间：①缺页率②磁盘读写时间③内存访问时间④执行缺页处理程序 CPU时间

170. inode有7个地址项，4个直接，2个1级，1个2级。地址项4B，块大小256B。单个文件最大长度 $=4\times256+2\times\frac{256}{4}\times256+1\times\frac{256}{4}\times\frac{256}{4}\times256=1057K$

171. 文件被用户进程首次打开时，OS需做的是：将FCB读入内存

172. 多个进程共享同一文件F，在系统打开文件表中仅有1个表项包含文件F属性。（系统打开文件表包含F属性）

173. 用于文件空闲块管理：位图/空闲磁盘块链/FAT（文件分配表）

174. inode原因：减少查找文件所需资源。

175. 逻辑地址实模式下由段基址+段内偏移组成。保护模式下由段选择符+段内偏移构成。不启用分页时：线性地址=物理，否则需转化

176. 什么是虚存？内存不足时以进程为单位在内外存之间交换。②对内存分页/分段管理之后，所有内存访问都是逻辑地址，运行时动态转为物理地址，使重定位成为可能，进程被划分成许多块，不需连续位于内存中。④进程不需所有页/段都在内存中（局部性原理，见下）

177. 时间局部性：某指令执行后不久可能再次执行，某数据访问后…
空间局部性：一段时间内访问的地址集中于一定范围（虚存）

178. 线程的模式切换比进程的开销低原因？包含的状态信息更少

179. 内存管理需求：重定位、保护、共享、逻辑组织、物理组织

180. 驻留集和工作集：前者指当前主存中这个进程页的个数，后者指这进程最近使用的页的个数

181. 执行I/O的技术：①可编程I/O：处理器代表进程发送I/O命令，进程进入忙等待，等待操作完成 ②中断驱动I/O：处理器代表进程向I/O模块发送I/O命令，然后继续执行后续指令，当模块完成工作后，处理器将该模块中断。如果该进程不用等I/O完成，则后续指令仍是该进程的指令，否则该进程在此中断挂起，处理器执行其他工作

控制主存和I/O的数据交换，为传送这数据，处理器给DMA发请求，只有当数据块整块传完后才中断。

181. 倒排页表：虚拟页号通过一个散列函数映射到散列表中，散列表包含指向倒排表的指针。不论有多少进程，多少虚拟页，页表只需实存中的一个固定部分

182. OS发展 ①串行处理，一次装入并运行1个作业 ②简单批处理，同时装入多程序，串行执行 ③多道程序批处理：同时装入多程序，并发执行，仅当I/O~~切换~~等待时切换 ④分时处理，③+执行时间长 ⑤对称多处理

183. 调度算法汇总：
FCFS：公平、实现简单/不利于短作业
RR：兼顾作业长短/平均等待时间长，上下文切换浪费资源
SPN：非抢占，平均等待时间最少/长进程会饥饿
SRT：抢占（进程到达时）周转时间高于SPN/长进程会饥饿
HRRN：非抢占，兼顾长短作业/需计算响应比的开销
反馈：抢占（时间片用完时）兼顾长短作业，有好的响应时间

184. 操作系统内核典型功能：①进程管理②内存管理③I/O管理④支持功能

185. 混合型线程：①线程创建在用户空间完成 ②线程的同步和调度在程序中进行 ③1个进程的多个用户级线程会映射到内核级线程上，同一应用程序的多个线程可在多个处理器上并行 ④引起阻塞的系统调用不会阻塞整个进程

186. 比较和交换指令

```
int bolt; void p(i){while(true){
while(compare&swap(bolt,0,1)==1);
临界区 bolt=0; 其余部分;}}
bolt=0; parbegin(p(1)...p(n));
```

187. 信号量令为1，写读优先

```
reader(){while(true){sW(z);
sW(rsem);sW(x);rc++;if(rc==1)
sW(wsem);sS(rsem);sS(x);
READ();sW(x);rc--;if(rc==0)
sS(wsem);sS(x);}
writer(){while(true){sW(y);
wc++;if(wc==1)sW(rsem);
sS(y);sW(ws);WRITE();sS(wsem);
sW(y);wc--;if(wc==0)sS(rsem);
sS(y);}} rc=wc=0;
```

188. 到达时间/服务时间 A 0/3 B 2/6 C 4/4 D 6/5 E 8/2
反馈 q=1：AABACBDCEDEBCDBCDBDB
$q=2^i$：AABACBBDECCDDEBBBCDD

189. 保护模式下，段寄存器仍16位，提供不了段基址，16位段寄存器（CS,DS,ES,FS,GS,SS）中不存段基址而是段选择符（在段描述符表中）

190. GDT、IDT系统只有1张，每个任务都有私有的LDT
GDT：系统使用的代码段、数据段、堆栈段和特殊数据段描述符，及所有的LDT的描述符。
LDT：本任务涉及的代码、数据、堆栈…描述符
IDT：中断向量表

191. 虚存手段：页管理、部分加载、按需调页、换入换出

192. 文件系统在外存中结构：
主引导区—分区表—分区1、2—2—3
分区1：引导区—超级块—空闲区管理—inode—根目录区—文件

193. 分时系统实现RR需：①PCB且 ②时钟中断处理程序 ③进程
绪队列

194. 可引起进程调度程序执行：
中断处理结束②阻塞③进程结束④进时间片用完

195. 动态重定位→作业执行过程

196. 虚存：访问页面不在主存中，
可用页框，则处理顺序：中断—
决定淘汰页→页面调出→调