2. 进程概念

①进程映像（Peocess Image）由程序代码（Programm Code）、相关数据（Set of Data）、进程控制块（Process Control Block）组成；可执行文件被映射进虚拟存储空间加上堆栈和PCB成为进程映像；

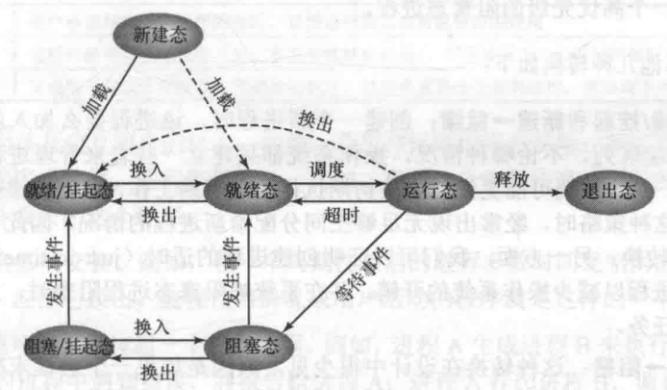
②可执行文件会被映射到虚存中，因此被称为映像文件；映像文件在虚存中被称为可执行文件映像；

1. 进程控制块的组成

①标志信息（标志符）：进程标志符Process Identification(PID)、父进程标志符（PPID）、用户标志符（UID）

②处理器状态信息：用户可见寄存器、控制和状态寄存器（程序计数器、条件码EFLAGS、状态信息）、栈指针（SP）

③进程控制信息：调度和状态信息（进程状态、优先级、调度相关、事件）、数据结构、进程间通信、进程特权、存储管理、资源分配；

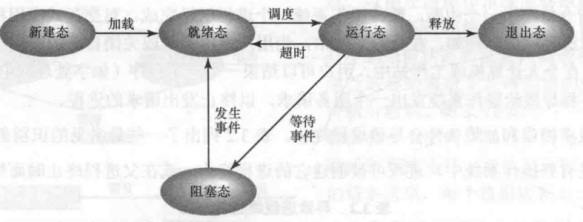
1. 进程状态及模型

运行→就绪：超时；抢占（Preempted）；主动释放；

解决I/O速度慢，所有进程等待I/O

就绪→就绪/挂起：优先挂起阻塞态；高优先级阻塞将就绪；

阻塞/挂起→阻塞：少见，极高优先级阻塞/挂起将就绪；



1. 进程创建

①为新进程分配一个唯一进程标志符，并申请一个空白PCB；②为进程分配资源，资源不足会阻塞；③初始化PCB，如标志、状态等；

④加入就绪队列或挂起队列；

⑤进程创建事件：用户登录（分时）；作业调度（调入）；提供服务（用户请求）；应用请求（fork）；

1. 进程终止

①从PCB集合中找到终止过程PCB；②若在运行，立即剥夺CPU；③终止所有子进程；④将该进程所有资源归还给父进程或系统；⑤删除PCB；

⑥进程终止事件：正常结束（包括程序报错自行退出）；异常结束（不可恢复错误）；外界干预（信号）；

1. 进程切换

①概念：上升/下降进程（调出/调入CPU进程）；现场（通用寄存器和状态字寄存器EFLAGS）；断点（程序计数器）；中断栈（内核栈，PCB或task\_struct中存在指向内核栈的指针）；

②进程切换的事件：中断、陷阱（系统调用）

③进程映像的组成：

用户级上下文：程序块、数据块、运行时堆和用户栈、共享内存；

寄存器上下文：程序计数器（PC）、状态寄存器（EFLAGS）、栈指针（SP）、通用寄存器； （硬件级上下文）

系统级上下文：进程表项（PCB）、内核栈、其他数据结构；

1. 在进程上下文切换时，操作系统把下降进程的寄存器上下文保存在系统级上下文的PCB和内核栈中（Linux中，通用寄存器在内核栈）
2. CPU处理中断过程

①关中断，防止新中断破坏现场和断点保存；②保存用户栈中的栈指针到内核栈；③栈指针指向内核栈；④保存EFLAGS到内核栈；⑤保存程序计数器到内核栈；⑥程序计数器指向中断处理程序第一条指令；⑦执行中断处理程序，将所有寄存器保存到内核栈；

1. 堆栈切换

由用户态切换到内核态时，当前栈必须切换到内核栈；当前栈时SP的值所在的栈空间，如果SP的值位于用户栈范围内，当前栈就是用户栈；寄存器SS的值应当指向当前栈的栈顶；每个进程都有自己的内核栈，PCB中有指向内核栈的指针；切换过程中，先将SS、SP压入内核栈再指向内核栈，然后压入EFLAGS、CS、PC；

1. 进程切换步骤

①保存硬件级上下文（寄存器上下文）；②修改PCB并挂载到相应队列（系统级上下文）；③调度另一个进程；④修改所选进程的PCB（系统级上下文）；⑤更新内存管理数据结构（系统级上下文）；⑥将硬件级上下文恢复为所选进程上次退出运行态时的上下文；

2. 线程的基本概念

引入进程是为了使多道程序并发执行，提高资源利用率和系统吞吐量；引入线程是为了减小进程在并发执行时所付出的时空开销，提高操作系统并发性能；线程状态：运行态、就绪态、阻塞态；线程组成：线程ID、程序计数器、寄存器集合、堆栈、优先级等；

1. 进程和线程的区别

引入线程的操作系统中，线程是独立调度的基本单位，进程是拥有资源的基本单位；（调度）

线程不拥有系统资源，但进程的各线程间共享进程资源；（拥有资源）

线程间可并发执行，系统有更好的并发性和更大的吞吐量；（并发性）

创建或撤销进程时系统开销远大于创建或撤销线程时，线程切换时开销小，且线程共享地址空间同步和通信容易；（系统开销）

进程地址空间相互独立

1. 用户级线程
2. 系统级线程
3. 组合方式
4. 多线程模型
5. Linux线程

8. 内存管理（Memory Management）
9. 内存地址

①内存地址从0开始，每个地址对应一个存储单元；②按字节编址，存储单元大小为1字节；按字编址，存储单元大小为1字；③地址长度就是地址空间需要用多少个二进制位表示；

1. 内存管理的功能

①内存空间的分配与回收；②地址转换；③内存空间的扩展（虚存）；④内存保护（进程互不干扰）；

1. 地址绑定时间

①程序设计时，物理地址；②编译汇编时，符号地址访问；③加载时，加载器转换；④运行时，硬件转换；

1. 链接的三种方式

①静态链接 ：符号解析（查符号表找模块）和重定位（符号引用变相对地址）在链接时全部完成，装载模块不含任何未解析符号引用；

②装入时动态链接：加载模块含到其他模块的未解析引用，加载时任何对其他模块的符号引用导致加载器查找加载该模块，并重定位；

③运行时动态链接（Run-Time Dynamic Linking）：加载到内存的模块页面中含有未解析引用，运行到该页时会符号解析并重定位；

1. 装入内存的三种的方式

①绝对加载：

定义：链接器生成的加载模块使用物理地址（程序员或编译器给出），模块总被加载到同一位置；

实现：用符号表示内存访问地址，编译或汇编时解析符号转换为地址，输入到绝对加载器；

缺点：加载前把内存访问绑定到具体地址，使加载模块只能放在同一位置，只适合单道程序；

②可重定位加载（静态重定位）：

定义：加载模块使用相对地址，模块起点地址为0，模块可以加载到不同位置后不变；

实现：加载模块包含内存始址等重定位地址库，由编译器生成；

缺点：加载时内存访问被绑定到绝对地址，换入换出不经过加载器，不能换到不同位置；

③动态运行时加载（动态重定位）：

定义：加载后运行前使用相对地址，模块在内存中位置运行时可变；

实现：加载模块加载到内存后，内存访问为相对地址，运行时由处理器计算物理地址（效率）；

特点：不连续存储区；装入部分代码运行；动态申请内存；实现虚拟存储；

1. 逻辑地址（Logical Address）空间与物理地址（Physical Adress）空间

①逻辑地址是一个字相对于程序始址0的距离/位置，链接后文件地址从0开始，称为逻辑地址空间；

②相对地址是逻辑地址特例，相对段/页始址的地址；

③将逻辑地址转换为物理地址称为地址重定位；

1. 内存保护

操作系统不受用户进程影响，用户进程不受其他用户进程影响。

①CPU中设置上、下限寄存器，存放用户作业上下限地址，访问地址时比较；

②重定位寄存器（基址寄存器）含最小物理地址，界地址寄存器（限长寄存器）含逻辑地址的最大值；将逻辑地址与界地址寄存器比较，未越界，则与重定位寄存器相加，得到物理地址。

1. 覆盖(Overlaying)

定义：将用户空间分成一个固定区和若干覆盖区，活跃部分放在固定区，其余部分按调用关系分段，先访问的放在覆盖区，其他段放在外存；

特点：打破进程全部信息放入主存才能运行的限制；但同时运行代码段大于主存不能运行；内存只能更新覆盖区；必须声明覆盖结构；

1. 交换

定义：将进程还如内存和换出内存；交换技术可用于两个及以上进程，覆盖技术用于单个进程；

1. 交换区：磁盘空间被分为文件区和交换区，文件区追求存储空间利用率，采用离散分配方式；交换区追求换入换出速度，采用连续分配方式；数据换出时放在交换区；
2. 连续分配管理方式——单一连续分配

定义：内存分系统区和用户区；

优点：简单；无外部碎片(External Fragmentation)；覆盖技术；无须内存保护；

缺点：只能用于单用户、但任务系统；有内部碎片（分配给进程的内存区域，有部分没有利用）；

1. 连续分配管理方式——固定分区分配

定义：将用户空间划分为多个固定的大小的分区，每个分区装入一个进程；有分区大小相等和分区大小不等两类；分区按大小排队，建立分区说明表，表项包括分区始址、大小、状态；

优点：用于多道程序设计最简单的存储分配；无外部碎片；

缺点：程序可能太大方不仅任何分区；存在内部碎片(Internal Fragmentation)，内存利用率低；

1. 连续分配管理方式——动态分区分配

定义：分区长度数量可变，进程装入内存，系统分配与其所需容量相等的内存空间；采用空闲分区表或空闲分区链记录；

缺点：会产生大量外部碎片（空闲分区太小而难以利用），可以通过紧凑（Compaction）技术解决；

动态分区分配策略：①首次适应（First Fit）算法：空闲分区地址递增顺序链接，顺序查找第一个满足要求的分区；②最佳适应（Best Fit）算法：空闲分区容量递增顺序链接；③最坏适应（Worst Fit）算法：最大适应算法，空闲分区容量递减顺序链接；④邻近适应（Next Fit）算法：循环首次适应算法，从上次查找结束位置继续查找；

1. 连续分配管理方式——伙伴系统

定义：可分配内存空间是大小为2^U个字的块；若请求大小s满足2^U-1<s<=2^U则分配整个空间，否则分成大小均为2^U-1个字的伙伴；反复进行直到找到大于等于s的最小块；合并时类似2048；

1. 非连续分配管理方式——基本分页存储管理

定义：把内存空间划分为大小相等且固定的块，称为页框(Frame)或页帧；将进程逻辑地址空间划分成与页框大小相等的数据块，存储在外存可临时复制到页框，称为页（Page）或页面；（页框=页帧=内存块=物理块=物理页面；页框号=页帧号=内存块号=物理块号=物理页号；页=页面；页号=页面号）；

分页与固定分区：一个程序可占多个分区；分区不需要连续；

页表(Page Table)：①页表记录进程页面对应内存页框位置，一般放在PCB中；②一个进程一个页表；③进程每个页面对应一个页表项（PTE, Page Table Entry）；④页表项由页号和块号组成；⑤倒排页表：用页框号作为页表索引（顺序排列），散列表映射虚存；

地址结构：逻辑地址结构由页号和逻辑偏移量组成；物理地址结构由页框号/内存块号和页内偏移量组成；如果K位表示页内偏移量，则页面大小为2^K个字节；如果M位表示页号，则进程最多有2^M个页面；

地址变换：①计算页号（逻辑地址/页面大小）和页内偏移量（逻辑地址%页面大小）；②比较页号和页表长度，判断越界；③计算页表项地址（=始址+页号\*页表长度），取内存块号；④块号和页内偏移得物理地址；

（页表长度表示有多少页表项，页表项长度表示页地址占多大存储空间）

例1：内存大小4GB，页面大小4KB，每个页表项多少字节？

①页帧大小=页面大小=4KB=2^12B；②4GB内存被分成2^32/2^12=2^20个内存块；③块号至少用20bit表示；④按字节编址，至少用3B表示；（页表项大小相等连续存放与逻辑地址对应，因此页号隐含不占空间）

1. 非连续分配管理方式——有快表的分页存储

定义：快表放在转换检测缓冲区（TLB，Translation Lookaside Buffer）；硬件允许同时查询多个TLB表项称为关联映射（Associative Mapping）；

单级页表问题：①根据页号查询页表要求页表项必须连续存放，当页表很大时要占用连续多个页框；②无须整个页表常驻内存，可能只要几个页面；

地址变换：①将逻辑地址分为页号和页内偏移；②将页号与TLB中页号比较；③若页号匹配，则页表项在快表中，取出页框号，与偏移量拼接成物理地址；④若未匹配，访问内存页表，读取页表项后，复制一份到快表；

局部性原理（Principle Of Locality）：时间局部性原理是执行过的指令不久后会再次执行；空间局部性原理被访问存储单元附近的存储单元也会被访问；

1. 非连续分配管理方式——使用两级页的表分页存储

定义：由于单级页表过大，通常将页表保存在虚存中像页面一样服从分页管理，运行时将一部分包括正在运行页的页表放在内存；

逻辑地址空间格式：一级页号、二级页号和页内偏移组成；n级页表对内存访问n+1次；

地址变换：逻辑地址拆成三部分；从PCB中读取页目录表始址，根据一级页号查页目录表找到下级页表的块号；根据二级页号查找二级页表找到内存块号；结合页内偏移找到物理地址，访问；

1. 非连续分配管理方式——基本分段存储管理方式

定义：这里的段指Section是变长数据块，在Linux链接过程中会合并为Segment；

分段与动态分区：一个程序可占多个分区；分区不需要连续；

逻辑地址空间格式：由段号和段内偏移量组成；段号的位数决定每个进程最多多少段；段内地址位数决定段最大长度；

段表：每个进程一个段表；每个段表项对应进程的一段；段表项包括段号、段长和起始地址；段号隐含；

地址变换：①将逻辑地址分成段号和段内偏移量；②比较段号和段表长度，判断越界；计算段表项地址，取出段长与段内偏移量比较，判断越界；④从段表现取出始址，与偏移量结合得到物理地址；

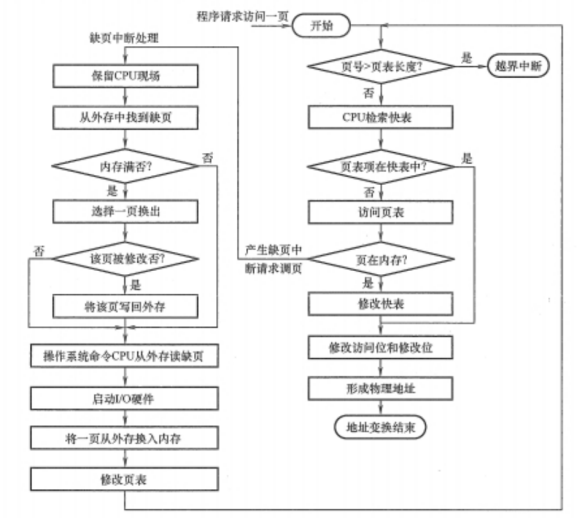
段的共享与保护：两个进程段表中相应的表项指向共享段的同一个物理副本；分页管理中，只需判断页号是否越界，页偏移不会越界，而段偏移大于段长会产生越界；

1. 非连续分配管理方式——段页式管理方式

分页和分段特点：分页管理内存利用率高，无外部碎片，只有少量内部碎片；难按逻辑模块实现共享保护；分段管理方便逻辑模块实现共享保护；段长过大难以分配连续空间，有外部碎片；

逻辑地址空间格式：包括段号、页号和页内偏移量；每个进程一个段表，每个分段一张页表；段表寄存器记录段表始址和段表长度；

地址变换：①逻辑地址的得到段号、页号、页内偏移；②判断段号越界；③段表寄存器和段号段表项长度得到段表项及其中页表始址；③检查页号越界；④根据页表始址、页号和页表项长度得到页表项及其中内存块号；⑤结合得到物理地址；

1. 虚拟内存（待补充）
2. 传统存储管理特征

①一次性：一次性装入内存才能运行；不能全部装入不能运行；大量作业少数运行；

②驻留性：装载后驻留内存；

1. 虚拟存储器特征：

多次性：无须一次性装入，进程只放入某些页面从而放置更多进程，保证至少一个就绪；

对换性：作业无须常驻内存，可换入换出；

虚拟性：逻辑上扩充容量，看到容量大于实际；

1. 虚拟内存和传统存储区别：

请求调页：进程访问内容不存在内存，操作系统将缺页从外存调入内存；

页面置换：暂时用不到的页面换出内存；

1. 请求分页管理

页表项：包括页号、块号、状态位（是否调入内存）、访问字段（一段时间内被访问次数、本页多长时间未被访问）、修改位（调入内存后是否被修改）、外存地址；

缺页中断：内部中断，一条指令执行期间可能有多次缺页中断；

地址变换：如图；

1. 页面置换算法

最佳置换算法（OPT）：换出以后永不使用或最长时间未使用的页面，最低缺页率（缺页中断次数/访问页面次数）；

先进先出页面置换算法（FIFO）：换出内存中驻留最久最早进入的页面，会产生Belady异常；

最近最久未使用置换算法（LRU,Last Recently Used）：换出最近最长时间为访问过的页面，需要硬件；

简单时钟置换算法（Clock）：页面首次装入内存使用位置1，每次被访问置1；换出时指针经过的页面置0，找到第一个0；

改进时钟置换算法（Clock）：①设状态位和修改位，有如下组合：未访问，未修改（0，0）；未访问，被修改（0，1）；被访问，未修改（1，0）；被访问被修改（1，1）；②第一次扫描不修改，换出第一个（0，0）；第二次扫描换出第一个（0，1），经过一页修改位置0；第三次扫描重复第一次；第四次扫描重复第二次；

Belady异常：分配物理块数量增大缺页率上升的现象；

页缓冲：LRI复杂开销大，VAX VMS方法使用FIFO，换出页面未修改放到空闲页链表，被修改放到修改页链表，只修改页表项，页还在内存中。空闲页链表和修改页链表充当高速缓存的角色；

1. 页面分配策略(驻留集管理)

驻留集：给一个进程分配的物理页框的集合；

固定分配（Fixed-Allocation）局部置换：每个进程分配一定数量物理块保持不变，缺页只能从驻留集中换出（不能确定数目）；

可变分配全局置换(Global Replacement)：系统保持空闲物理块队列，进程缺页就分配（盲目增加物理块，并发能力下降）；

可变分配(Variable-Allocation)局部置换(Local Replacement)：每个进程一定数目物理块，驻留集根据缺页率变化；

1. 调入页面时机

预调页（Prepaging）策略在运行前调入，请求调页（Demand Paging）策略在运行期间调入；

1. 调入页面位置

有足够对换区空间：运行前将文件调入交换区，全部从交换区调入内存；

缺少足够交换区：不会被修改的文件从文件区调入，被修改过的文件换出到交换区；

UNIX方式：未运行过的页面从文件区调入，换出页面在交换区；

非一致存储访问（NonUniform Memory Access, NUMA）多处理器：访问存储物理单元时间与处理器到内存模块间距离不同变化；

1. 抖动（Threashing）

频繁换入换出的现象称为抖动或颠簸（处理器大部分时间用于交换而非执行命令），主要原因是频繁访问页面数目高于分配页帧数目；

1. 工作集策略（Working Set Strategy）

某段时间间隔内，进程要访问的页面合集，用最近访问过的页面代替；工作集由时间和工作集窗口大小确定；

1. Linux内存管理

虚存：Linux使用三级页表结构（页目录、页中间目录、页表、偏移量）；

页面分配：为把连续页分配到连续页框，内核维护大小固定连续页框组（1、2、4、...、32个）使用伙伴算法分裂合并；

页面置换算法：分割LRU算法；