# 1.温故而知新——操作系统概念

**北桥：**连接高速芯片

**系统调用接口：**以软件中断的方式提供，如Linux使用0x80号中断作为系统调用接口

**多任务系统：**进程隔离

设备驱动

**直接使用物理内存的弊端**

地址空间不隔离

内存使用效率低

程序运行的地址不确定

分段

分页

**线程：**轻量级进程LWP，是程序执行流的最小单元。一个标准的线程由线程ID，当前指令指针PC，寄存器集合和堆栈组成。通常意义上，一个进程由一个到多个线程组成，各个线程之间共享程序的内存空间（包含代码段、数据段、堆等）及一些进程级的资源（如打开文件和信号）

CPU密集型（很少等待的线程）和IO密集型（频繁等待的线程）

优先级调度下，线程的优先级改变的方式

用户指定优先级

根据进入等待状态的频繁程度提升或降低优先级

长时间得不到执行而被提升优先级

线程安全

原子操作

同步和锁

**信号量：**允许多个线程并发访问，可以被任意线程获取并释放

**互斥量：**谁获取，谁释放

**临界区：**只有创建的进程可见。

读写锁

条件变量

**被重入：**一个函数被重入，表示这个函数没有执行完成，由于外部因素或内部调用，又一次进入该函数执行。一个函数被重入的两种情况：

多个线程同时执行这个函数

函数自身（可能经过多层调用之后）调用自身

一个函数要成为可重入函数，必须具有的特点：

不使用任何（局部）静态或全局的非const变量

不返回任何（局部）静态或全局的非const变量的指针

仅依赖于调用方提供的参数

不依赖于任何单个资源的锁（mutex等）

不调用任何不可重入的函数

可重入是并发安全的强力保障，一个可重入的函数可以在多线程环境下放心使用。

**过度优化：**volatile关键字试图阻止过度优化，它可以做两件事

阻止编译器为了提高速度将一个变量缓存到寄存器内而不写回

阻止编译器调整操作volatile变量的指令顺序。

# 6.可执行文件的装载与进程

**动态装载方法：**原则上利用程序的局部性原理，思想是程序用到哪个模块就将哪个模块装入内存，如果不用久暂时不装入，存放在磁盘中。

**覆盖装入：**由程序员编写管理器分配内存，决定哪些内存可以驻留和丢弃。在没有虚拟内存之前使用，现在此方法已废弃，除了嵌入式开发

**页映射：**当前的装载方式。

**进程的建立：**

创建一个独立的虚拟地址空间。在Linux上，就是分配一个页目录。

读取可执行文件头，并且建立虚拟地址空间与可执行文件的映射关系。Linux把进程虚拟空间的一个段叫做虚拟内存区域（VMA）。

将CPU的指令寄存器设置成可执行文件的入口地址，启动执行

随着程序的执行，页错误会不断地发生，操作系统也会为进程分配相应的物理页面来满足进程执行的需求。

对于相同权限的段，把它们合并到一起当做一个段进行映射。

查看不同段的映射情况 readelf -l /bin/cp

**关于可执行文件装载:**

加载器读取可执行文件头后,会分析可执行文件中的段表信息,根据不同段的属性和权限要求,将它们映射到进程虚拟地址空间的不同区域。

对于只读数据段,加载器会与相同属性的其他段合并映射,实现共享。对于代码段,通常也会共享映射。

对于可写数据段,加载器会分配独立的物理内存空间,实现数据的隔离。栈区也需要独立映射。

**关于进程创建:**

除了设置指令指针,加载器还会初始化进程的栈区,并根据可执行文件头信息设置 argc/argv 参数。

加载完成后,会设置进程状态为Ready,等待调度运行。

内核会为新进程分配一个独立的PID,并在内部进程表中注册该进程。

首次运行时,会执行初始化/设置函数(如libc\_start\_main)做进一步初始化。

进程并不随加载器而终止,也可以执行动态链接来加载依赖库。

进程在运行时会受到内存隔离保护,无法访问其他进程的内存空间。