# 操作系统

进程是无法直接操作IO设备的,其必须通过系统调用来内核协助完成IP动作,而内核会为每个IO设备维护一个buffer.

整个请求为:用户进程发起请求,内核接受到请求后,从IO设备中获取数据到buffer中,再将buffer中的数据copy到用户进程的地址空间.

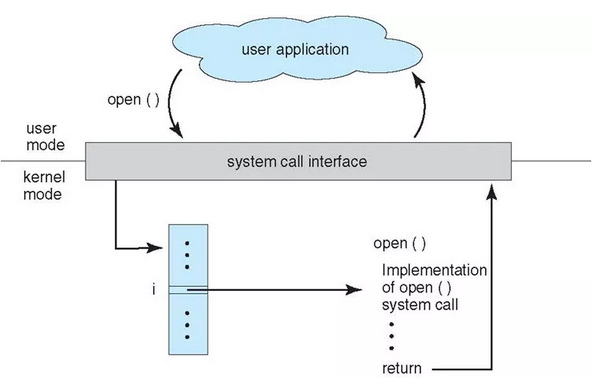
这里是用户进程请求内核把buffer的数据拷贝过来,这里就用到了系统调用read()函数了哦.

如果用户进程在read的时候buffer里面还没有数据,下面就是阻塞非阻塞啊,同步异步那套了.

## 系统调用

这里先说下系统调用,避免下面的不好理解.

如果一个进程在用户态需要使用内核态的功能,就进行系统调用从而陷入内核态,由操作系统代为完成.



Linux的系统调用主要有以下这些

| **Task** | **Commands** |
| --- | --- |
| 进程控制 | fork(); exit(); wait(); |
| 进程通信 | pipe(); shmget(); mmap(); |
| 文件操作 | open(); read(); write(); |
| 设备操作 | ioctl(); read(); write(); |
| 信息维护 | getpid(); alarm(); sleep(); |
| 安全 | chmod(); umask(); chown(); |

**看到了么,fork啊 sleep啊 read()啊这些操作系统的api其实都是系统调用.**

## 基本特征

并发 共享 虚拟 异步

## 作用

内存管理 文件管理 进程管理 设备管理

## 内核分类

大内核,大内核是将操作系统功能作为一个紧密结合的整体放到内核,由于各个模块共享信息,因此有很高的性能.

微内核,由于操作系统不断复杂,因此降一部分操作系统功能移出内核,从而降低内核的复杂性.移出的部分划分成若干的服务,相互独立.

微内核一个模块运行在内核态,其他的运行在用户太,这样需要内核态用户太经常切换,性能会低一些.

## 中断的分类

### 外中断

由cpu执行指令以外的事件引起,如I/O中断(包括键盘,鼠标啊,硬盘这些引起的中断),还有其他的时钟中断,控制台中断等.

### 异常

由CPU执行指令的内部事件引起,如非法操作码,地址越界,除0等.

### 陷入

用户程序中使用了系统调用.

## 进程与线程

进程与线程的区别,特定,状态.

调度算法.

临界区 同步与互斥 信号量

死锁 哲学家吃饭问题 银行家算法

似乎线程有很多优势，比如，数据共享效率高，可应对并发操作，有效利用等待时间等等，但是多线程的编程比多进程要复杂，同时，多进程的可靠性较好，因为进程间不会相互影响。实际情况还是需要自己分析拿捏的。但是一般来说，实际应用中常常采用“进程+线程”结合的方式，而不是非此即彼

## 进程间通信

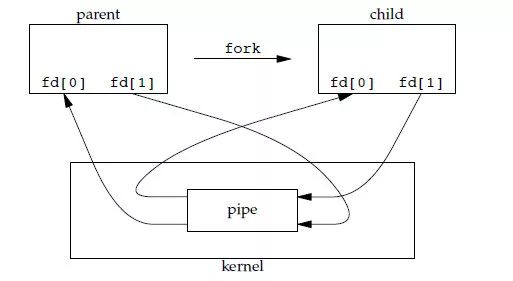
套接字是可以在不同机器进程中通信的方式!!!

### 管道pipe

这个有专门的api

#include <unistd.h>  
int pipe(int fd[2]);

只能在父子进程中使用,



### 消息队列

非常吊,好用.

### 信号量

这玩意只是一个计数器,用于为多个进程提供对共享数据对象的访问,

好像作用很小啊,就是个计数器啊,根本不如消息队列好用.

### 共享存储

允许多个进程共享一个给定的存储区.因为数据不需要再进程之间复制,所以这是最快的一个进程间通信.

以前我们知道进程是独立的拥有资源的单位,这个通信方式是让多个进程共享一块资源,所以这是一个特殊的方式,这个会借助信号量的帮助.

多个进程可以将同一个文件映射到他们的地址空间从而实现共享内存.

### 套接字

套接字可以用于不同机器之间的进程间通信.

## 内存管理

### 虚拟内存

虚拟内存的目的是为了让物理内存扩充成更大的逻辑内存,就是这个意思,操作系统把物理内存映射成逻辑内存,通过页关联起来,其实逻辑的内存比真正的物理内存要大,比如你的物理内存只有2g,但是逻辑内存可以映射出8g.

你的程序操作内存操作的是逻辑内存,比如你的这个程序要占用3g内存,那么你用的是逻辑内存的8g,但是呢,这样的话8g肯定无法映射到2g上啊,多啊,其实是这样的,一个程序在运行的时候并不需要把自己占用的逻辑内存全都加载到物理内存中,用到哪些逻辑页,把这些逻辑页加载到物理内存上就可以了.所以你的2g物理内存是可以搞出8g逻辑内存供应用程序使用的.

### 用户空间和内核空间

如果是4g的虚拟内存,对于linux操作系统来说,会有1g当作内核空间,保存内核的代码和数据,然后有3g分给应用程序使用,保存代码和数据.

外面的进程也可以通过系统调用进入内核,内核空间是所有进入内核的进程共享的,所以这样来看,应用进程可以使用的空间还是4g.

## 零拷贝

零拷贝描述的是CPU不执行拷贝数据从一个存储区域到另一个存储区域的任务，这通常用于通过网络传输一个文件时以减少CPU周期和内存带宽。

减少甚至完全避免不必要的CPU拷贝，从而让CPU解脱出来去执行其他的任务.

减少内存带宽的占用.

通常零拷贝技术还能够减少用户空间和操作系统内核空间之间的上下文切换.

上面我们也提到了,应用程序是不能操作内核中的给IO设备准备的buffer的,需要cpu把IO设备buffer中的内容复制到应用进程的地址空间里,零拷贝就是减少这步操作.

首先零拷贝这种东西一看就是要内核实现的东西,和java没有任何关系的.

通过sendfile实现的零拷贝I/O

带有DMA收集拷贝功能的sendfile实现的I/O

通过mmap实现的零拷贝I/O

上面这些都是linux实现零拷贝的机制.

select和epoll 函数监视的文件描述符分3类，分别是writefds、readfds、和exceptfds。

## BIOS

Cpu加电之后执行的第一条指令是在内存里,内存不是一断电就没了,不是的,内存分为RAM随机读存储,ROM只读存储,断电就没了的是随机读存储,ROM里面还是有以前写的内容的.

ROM里面就有一个地方存着BIOS的指令,cpu起来后就会去找这个指令,这个指令就会触发到把操作系统的程序(操作系统内核)加载到内存(这个程序叫bootloader),这样操作系统就起来了,然后控制权就给到了操作系统.

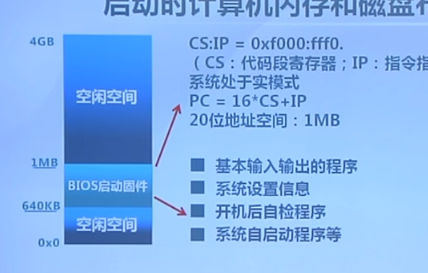
为啥BIOS叫基本输入输出系统呢,因为他要加载操作系统,操作系统内核是在磁盘里面的,所以BIOS必须要由输入输出的功能才可以呀!!!!

上面说了计算机是如何加载操作系统的,操作系统内核被加载进来后,他就是老大了,他也是被信任的东西了.但是还是需要中断和系统调用的.

中断解决,和外设硬件的联系,比如你键盘敲击了,我这我要用中断来弄.

还有就是软件出现异常了,也是中断的任务.

系统调用的是为软件提供服务的,操作系统不能让软件直接和cpu交互,不安全,不规范,只能对外提供接口,也就是系统调用来供应用程序调用.



DOS是一个操作系统,比较老的那种黑色命令行.

而BIOS是基本输入出入系统,通过这个基本输入输出命令,操作系统就是由BIOS拉起来的,也就是BIOS的运行不依赖于操作系统,操作系统不起来,不装BIOS都是可以执行的,可以修改操作系统的一些配置.