9.4系统调用

系统态、用户态：在实际运行过程中，处理机会在系统态和用户态间切换。相应的指令为：特权指令、非特权指令

系统调用：本质上是应用程序请求OS内核完成某功能时的一种过程调用

与一般调用的区别：运行在不同的系统状态、状态转换、返回、嵌套调用

中断机制：系统调用是通过中断机制实现的，并且一个操作系统的所有系统调用，都通过同一个中断入口来实现

进程控制类系统调用：创建和终止进程、获得和设置进程属性、等待某事件出现

文件操纵类系统调用：创建和删除、打开关闭、读写

进程通信类系统调用：打开连接、接受连接、发送消息、接收消息、关闭连接(参考进程通信)

POSIX标准：



9.5 UNIX系统调用

进程控制：创建进程fork、终止进程exit、等待子进程结束wait、执行一个文件exec、进程暂停(pause)、获取进程ID、获取用户ID

文件操纵：创建文件、 打开文件、关闭文件、读文件及写文件等二十多条（数量最多）

文件的读和写：文件描述符fd，缓冲区首地址buff，字节数nbyte（回忆计网课设的send）

进程通信：消息机制、共享存储器机制、信号量机制

信息维护：设置和获得时间、获得进程和子进程时间(times)、设置文件访问和修改时间(utime)、获得当前UNIX系统的名称(uname)

9.6 系统调用的实现

实现方法

1系统调用号和参数传递的设置：设置唯一的系统调用号、将系统调用号传递给中断和陷入机制，参数通过陷入指令自带方式、直接将参数送入相应的寄存器中、参数表方式等传递

2处理步骤：UNIX→check命令，MS-DOS→INT 21H

3调用处理子程序的处理过程

UNIX系统调用的实现

1CPU环境保护

2 AP和FP指针

**系统调用参数表指针AP：用于指示正在执行的系统调用所需参数表的地址**

**调用栈帧指针：用于指示本次系统调用所保存的数据项**

3确定系统调用号：

4参数传送

5利用系统调用定义表转入相应的处理程序

6系统调用返回前的公共处理

**重新计算该进程的优先级，发生了错误使进程无法继续运行时，系统会设置再调度标志**

若已设置**再调度标志**，便调用switch调度程序，再去从所有的就绪进程中选择优先级最高的进程，把处理机让给该进程去运行。

Linux系统调用

与unix类似

Linux系统在CPU的保护模式下提供了四个特权级别，目前内核都只用到了其中的两个特权级别，分别为“特权级0”(即内核态)和“特权级3”(即用户态)

Win32的应用程序接口(API)

API是一个函数的定义，说明如何获得一个给定的服务，而系统调用是通过中断向内核发出的一个请求

一个API函数可能不与任何系统调用相对应，也可以调用若干个系统调用

不同的API函数可能封装了相同的系统调用