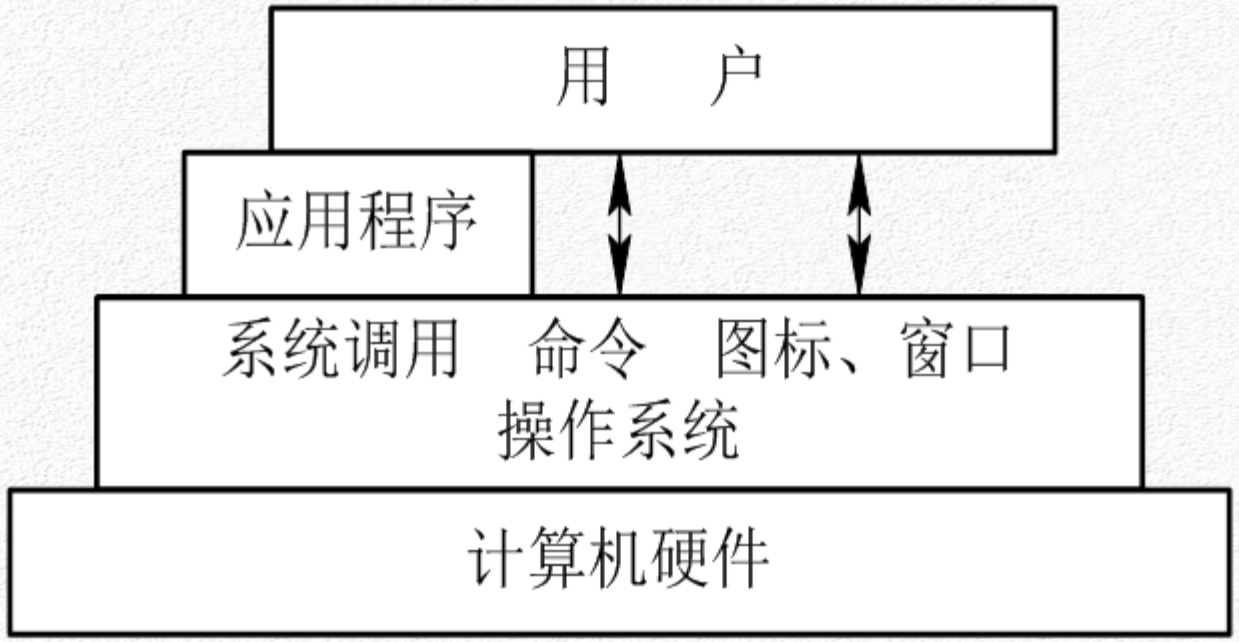
Linux操作系统编程

Linux文件的IO操作

Linux系统调用

●**系统调用：**操作系统提供给用户程序调用的一组“特殊”接口，用户程序可以通过这组“特殊”接口来获得操作系统内核提供的服务

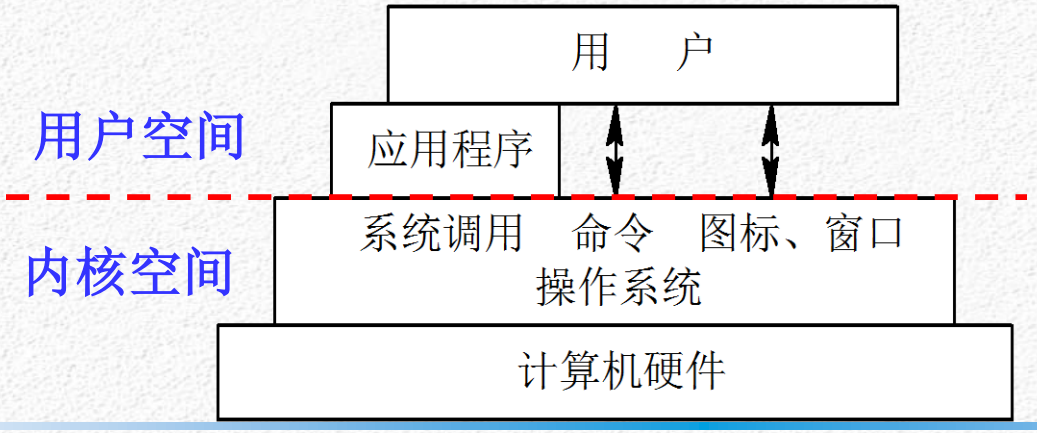


Linux系统调用

●为什么用户程序不能直接访问系统内核提供的服务?

- 为了更好地保护内核空间，将程序的运行空间分为内核空间和用户空间（也就是常称的内核态和用户态），它们分别运行在不同的级别上，在逻辑上是相互隔离的。因此，用户进程在通常情况下不允许访问内核数据，也无法使用内核函数，它们只能在用户空间操作用户数据，调用用户空间的函数。

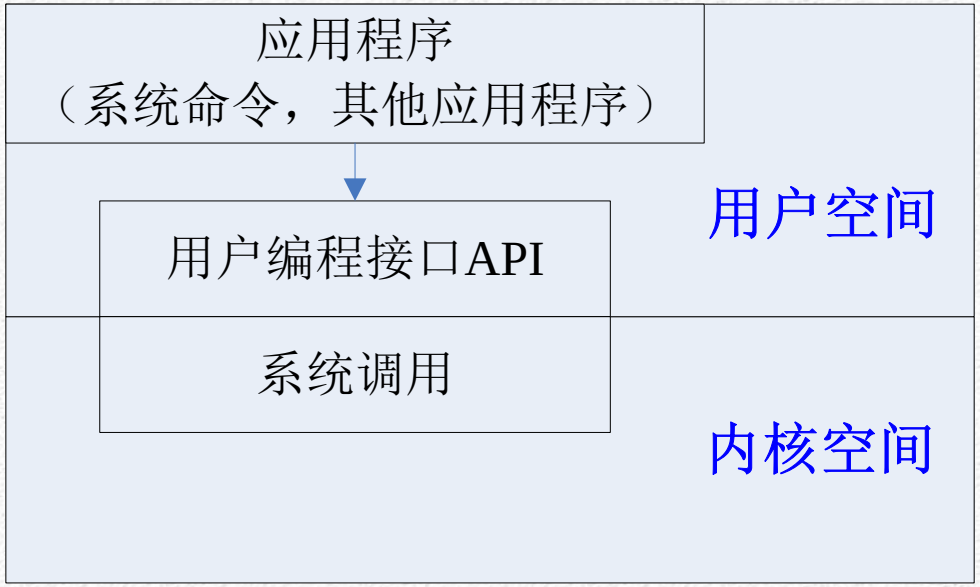
●进行系统调用时，程序运行空间从用户空间进入内核空间，处理完后再返回到用户空间



Linux系统调用

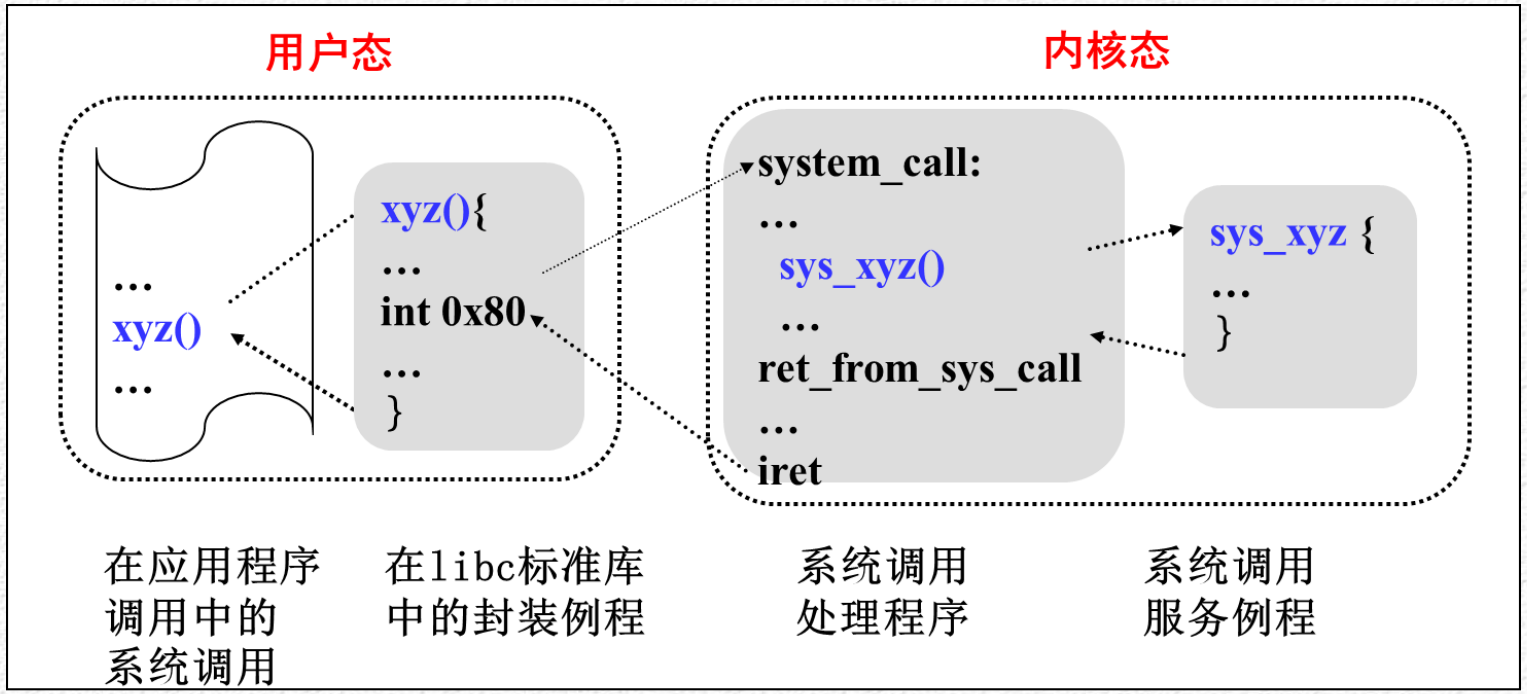
●系统调用并不是直接与程序员进行交互的，它仅仅是一个通过软中断机制向内核提交请求，以获取内核服务的接口。

●在实际使用中程序员调用的通常是用户编程接口——API。



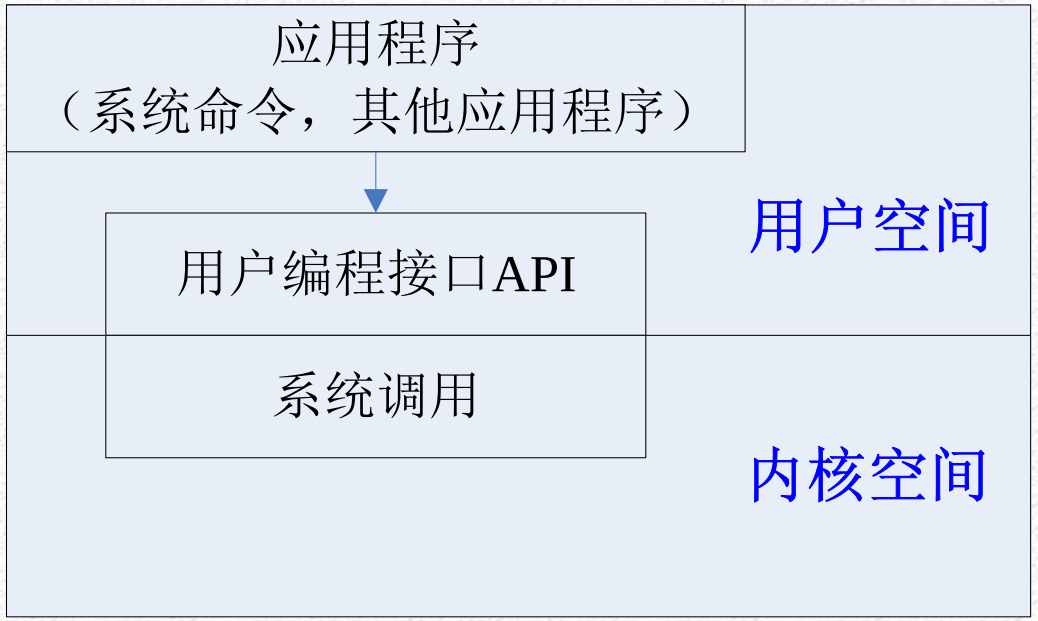
Linux系统调用

●Linux中的系统调用包含在Linux的libc库中，通过标准的C函数调用方法可以调用

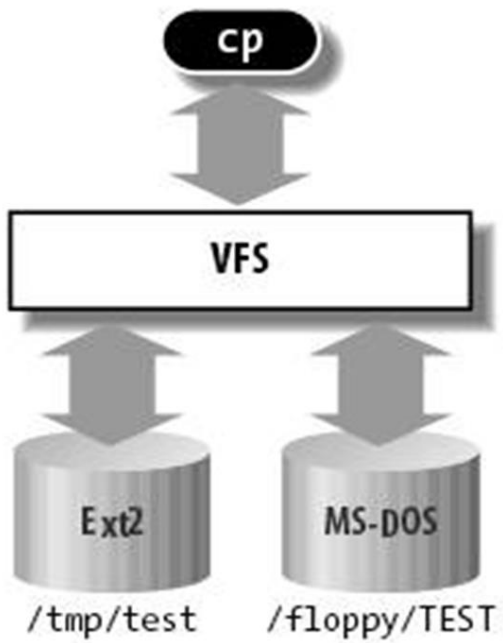


Linux系统调用

●系统命令相对API更高了一层，它实际上是一个可执行程序，它的内部调用了用户编程接口（API）来实现相应的功能。



- 【例】# cp /floppy/TEST /tmp/test



(a)

```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
       O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

(b)

文件描述符

●内核如何区分和引用特定的文件?

通过**文件描述符**。文件描述符是一个非负的整数，它是一个索引值，并指向在内核中每个进程打开文件的记录表。当打开一个现存文件或创建一个新文件时，内核就向进程返回一个文件描述符；当需要读写文件时，也需要把文件描述符作为参数传递给相应的函数。

●一个进程启动时，通常会打开3个文件：

-----标准输入　　描述符为0

------标准输出　　描述符为1

------标准出错处理　　描述符为2

Linux文件IO函数说明

- **open():**用于打开或创建文件，可以指定文件的属性及用户的权限等各种参数
- **creat():**打开一个文件，如果文件不存在，则创建它
- **close():**用于关闭一个被打开的文件。当一个进程终止时，所有被它打开的文件都由内核自动关闭，很多程序都使用这一功能而不显示地关闭一个文件
- **read():**用于将从指定的文件描述符中读出的数据放到缓存区中，并返回实际读入的字节数。若返回0，则表示没有数据可读，即已达到文件尾。读操作从文件的当前指针位置开始
- **write():** 用于向打开的文件写数据，写操作从文件的当前指针位置开始。对磁盘文件进行写操作，若磁盘已满或超出该文件的长度，则write()函数返回失败

Open函数

头文件

#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>

函数原型

int open(const char * pathname, int flags);
int open(const char * pathname,int flags, mode_t mode);

函数说明

参数pathname 指向欲打开的文件路径字符串。下列是参数flags 常用的标识:
O_RDONLY 以只读方式打开文件
O_WRONLY 以只写方式打开文件
O_RDWR 以可读写方式打开文件。上述三种标识是互斥的，也就是不可同时使用，但可与下列的标识利用OR(|)运算符组合。
O_CREAT 若欲打开的文件不存在则自动建立该文件。
O_TRUNC 若文件存在并且以可写的方式打开时，此标识会令文件长度清为0，而原来存于该文件的资料也会消失。
O_APPEND 当读写文件时会从文件尾开始移动，也就是所写入的数据会以附加的方式加入到文件后面。
O_NONBLOCK 以不可阻断的方式打开文件，也就是无论有无数据读取或等待，都会立即返回进程之中。
O_EXCL: 如果同时指定 O_CREAT，而该文件又是存在的，报错；也可以测试一个文件是否存在，不存在则创建。

返回值

文件打开成功返回文件的描述符，失败返回-1

open函数示例

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define FILE_PATH    "./test.txt"
int main(void)
{
    int fd;
    if ((fd = open(FILE_PATH, O_RDWR | O_CREAT | O_EXCL, 0666)) < 0)
      {
        printf("open error\n");
        exit(-1);
    } else {
        printf("open success\n");
    }
    return 0;
}
```