# 背景

# 概述

# 触发条件

用户态到内核态切换的触发条件：用户态需要申请外部资源

1. 系统调用
2. 中断
3. 异常

比如读写文件就会调用open、read、write系统调用，实现用户态和内核态的切换。

申请内存的malloc函数（不是系统调用），它有两种实现方式：小于128B的使用brk系统调用，大于128B的使用mmap内存映射。brk和mmap申请的是虚拟的地址空间，而不是真正物理内存空间。想要真正拿到物理内存空间，需要在第一次访问的时候触发缺页中断/缺页异常，实现用户态和内核态的切换。

# 系统调用

系统调用分为五类：

1. 进程控制相关：fork、exit
2. 文件管理相关：chmod、chown、open

注：fopen是库函数，open是系统调用，前者封装了后者

1. 设备相关：read、write、ioctl
2. 信息相关：getcpu、getxxx
3. 通信相关：pipe、mmap

# 通信方式

Linux用户态和内核态由于CPU权限的限制，通信并不像想象中的，使用进程间通信方式那么简单。

我们平常在写代码时，一般是在用户空间，通过系统调用函数来访问内核空间，这是最常用的一种用户态和内核态通信的方式。

除此之外，还有以下四种方式：

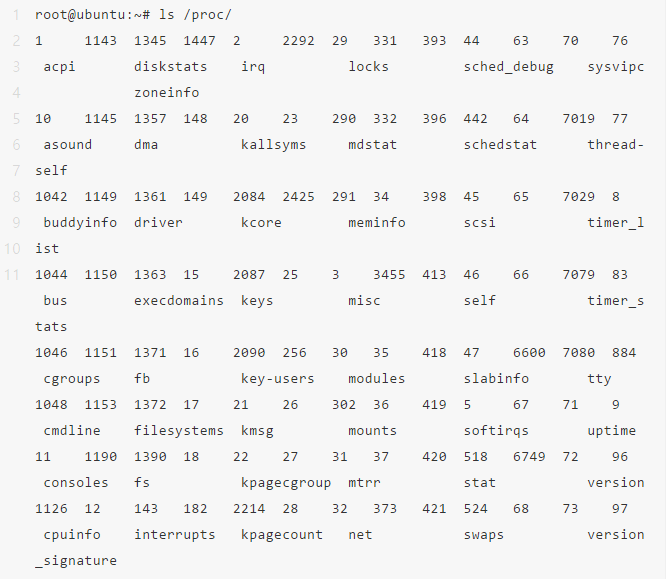
* procfs(/proc)
* sysctl(/proc/sys)
* sysfs(/sys)
* netlink套接口

## procfs(/proc)

procfs 是进程文件系统的缩写，它本质上是一个伪文件系统，为什么说是伪 文件系统呢？因为它不占用外部存储空间，只是占用少量的内存，通常是挂载在 /proc 目录下。

我们在该目录下看到的一个文件，实际上是一个内核变量。内核就是通过这个目录，以文件的形式展现自己的内部信息，相当于/proc 目录为用户态和内核态之间的交互搭建了一个桥梁，用户态读写 /proc 下的文件，就是读写内核相关的配置参数。

比如常见的 /proc/cpuinfo、/proc/meminfo 和 /proc/net 就分别提供了 CPU、内存、网络的相关参数。除此之外，还有很多的参数，如下所示：



可以看到，这里面有很多的数字表示的文件，这些其实是当前系统运行的进程文件，数字表示进程号（PID），每个文件包含该进程所有的配置信息，包括进程状态、文件描述符、内存映射等等，我们可以看下：

综上，内核通过一个个的文件来暴露自己的系统配置信息，这些文件，有些是只读的，有些是可写的，有些是动态变化的，比如进程文件，当应用程序读取某个 /proc/ 文件时，内核才会去注册这个文件，然后再调用一组内核函数来处理，将相应的内核参数拷贝到用户态空间，这样用户读这个文件就可以获取到内核的信息。一个大概的图示如下所示：

## sysctl

我们熟悉的sysctl是一个Linux命令，man sysctl可以看到它的功能和用法。它主要是被用来修改内核的运行时参数，换句话说，它可以在内核运行过程中，动态修改内核参数。

它本质上还是用到了文件的读写操作，来完成用户态和内核态的通信。它使用的是/proc的一个子目录/proc/sys。和procfs的区别在于：

procfs主要是输出只读数据，而sysctl输出的大部分信息是可写的。

例如，我们比较常见的是通过cat /proc/sys/net/ipv4/ip\_forward来获取内核网络层是否允许转发IP数据包，通过echo 1 > /proc/sys/net/ipv4/ip\_forward或者 sysctl -w net.ipv4.ip\_forward=1来设置内核网络层允许转发IP数据包。

同样的操作，Linux也提供了文件/etc/sysctl.conf来让你进行批量修改。

## sysfs

sysfs是Linux 2.6才引入的一种虚拟文件系统，它的做法也是通过文件/sys来完成用户态和内核的通信。和procfs不同的是，sysfs是将一些原本在procfs中的，关于设备和驱动的部分，独立出来，以“设备树”的形式呈现给用户。

sysfs不仅可以从内核空间读取设备和驱动程序的信息，也可以对设备和驱动进行配置。

我们看下/sys下有什么：

# ls /sys

block bus class dev devices firmware fs hypervisor kernel module power

可以看到这些文件基本上都跟计算机的设备和驱动等息息相关的。更多关于这些文件的解释大家可以自行了解，这里就不过多展开了。

## netlink

netlink是Linux用户态与内核态通信最常用的一种方式。Linux kernel 2.6.14 版本才开始支持。它本质上是一种socket，常规socket使用的标准API，在它身上同样适用。比如创建一个netlink socket，可以调用如下的socket函数：

#include <asm/types.h>

#include <sys/socket.h>

#include <linux/netlink.h>

netlink\_socket = socket(AF\_NETLINK, socket\_type, netlink\_family);

netlink这种灵活的方式，使得它可以用户内核与多种用户进程之间的消息传递系统，比如路由子系统，防火墙（Netfilter），ipsec安全策略等等。

引申：

net-tools工具通过procfs(/proc)和ioctl系统调用去访问和改变内核网络参数配置，而iproute2则通过netlink套接字接口与内核通信，前者已经被淘汰了，后者逐步成为标准。

# 总结

Linux 用户态和内核态通信主要的四种方式，其中netlink和procfs是最常见的方式。