北京郵電大學
Beijing University of Posts and Telecommunications

操作系统内核
-基于Linux

第7讲
系统调用

主讲：杨文川

内容

- 1 Linux中的各种API
- 2 系统调用的机制
- 3 系统调用的流程
- 4 系统调用的优化
- 5 动手实践-添加系统调用

内容导航：

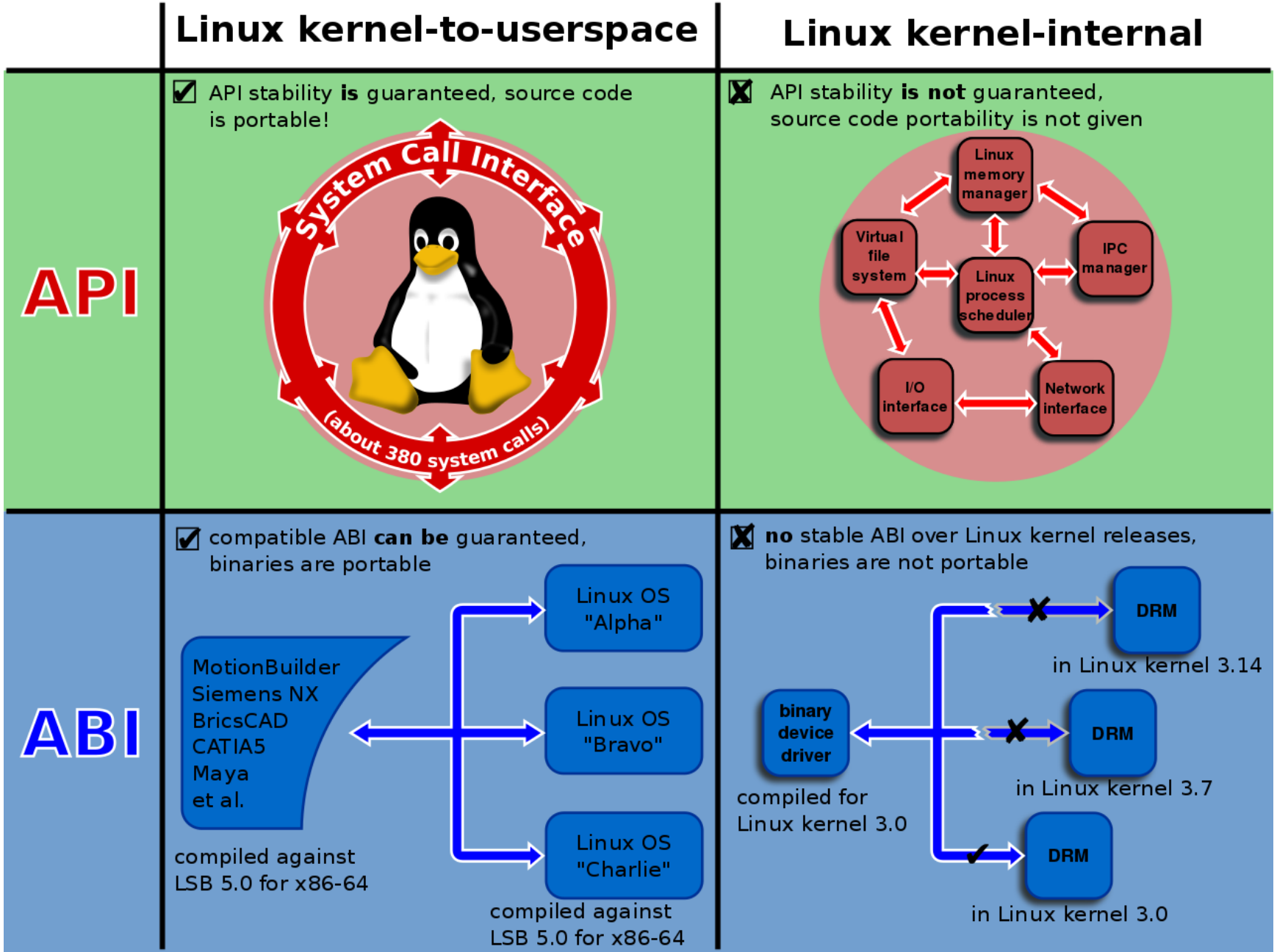
1 Linux中各种接口

Linux中各种接口

- 如果将内核比作一座工厂，那么Linux中众多的接口就是通往这个巨大工厂的高速公路。
- 这条路要足够坚固，禁得起各种破坏(Robust)。
- 要能跑得了运货卡车，还要能升降飞机(Compatible)。
- 当然了这条路要越宽越好(Performant)。
- 为此，Linux提供了多种接口来适应这些要求。

Linux中各种接口

- 如图所示，Linux中有四种类型的接口
- 应用编程接口(API)，
- 应用二进制接口(ABI)，
- 内核内部的API和ABI。
- 下面我们逐一的来看看这些接口。



LSB(Linux Standards Base)

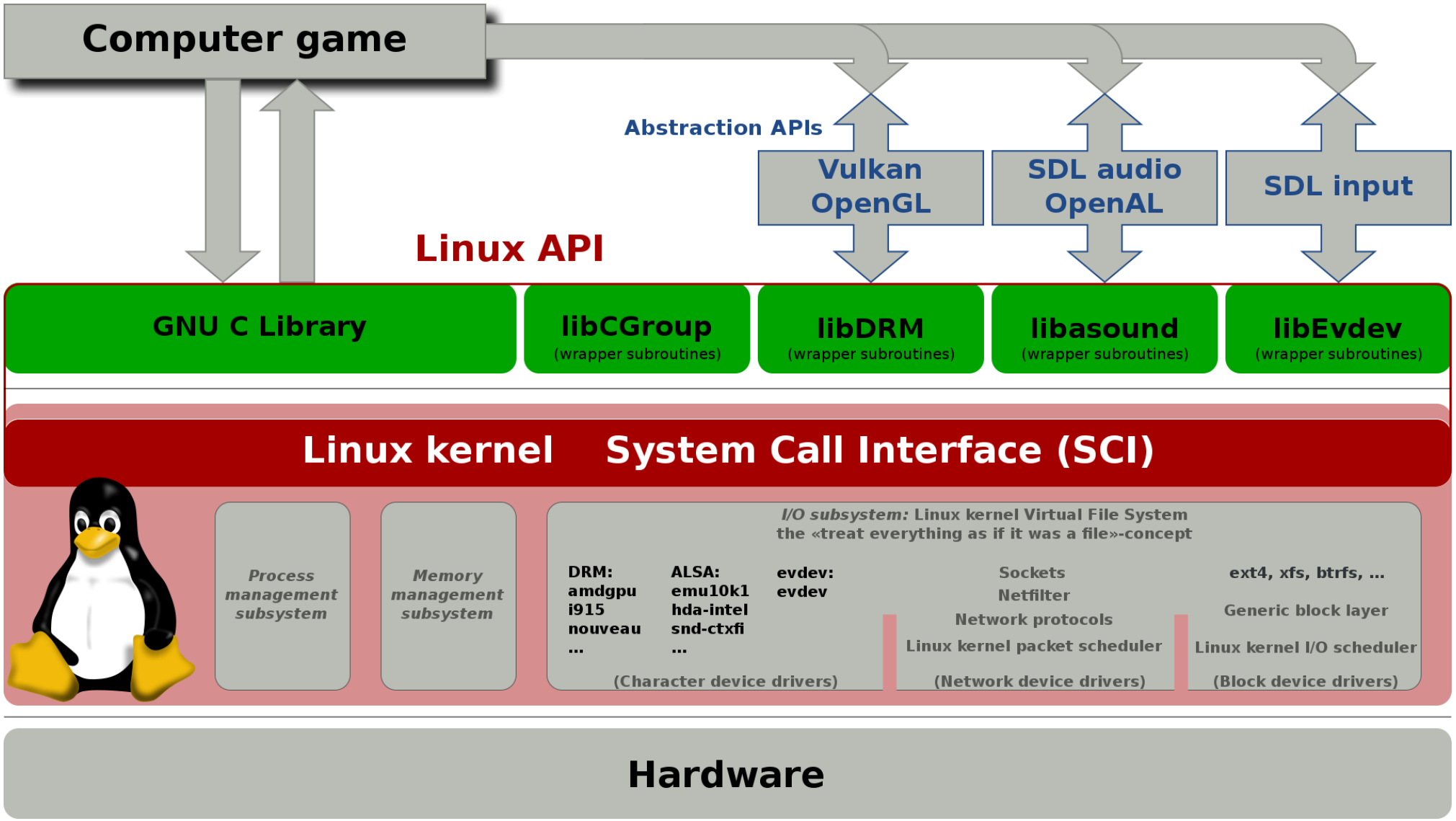
- LSB(Linux Standards Base)是什么？
- 目前Linux 的发行版非常繁多，为了促进Linux 不同发行版间的兼容性，LSB开发了一系列标准，使各种软件可以很好地在兼容LSB 标准的系统上运行，从而可以帮助软件供应商更好地在Linux 系统上开发产品，或将已有的产品移植到Linux系统上。

- LSB 是Linux 标准化领域中事实上的标准，它的图标形象地阐述了自己的使命：对代表自由的企鹅(Linux)制定标准。
- 给定企鹅的体形和三维标准之后，软件开发者就可以设计并裁减出各色花样的衣服(应用程序)，这样不管穿在哪只企鹅身上，都会非常合身。

Linux API

- 如图所示，主要有三部分：
- 1.Linux API
- 2.Linux内核系统调用接口SCI
- 3.C标准库

- 下面将详述



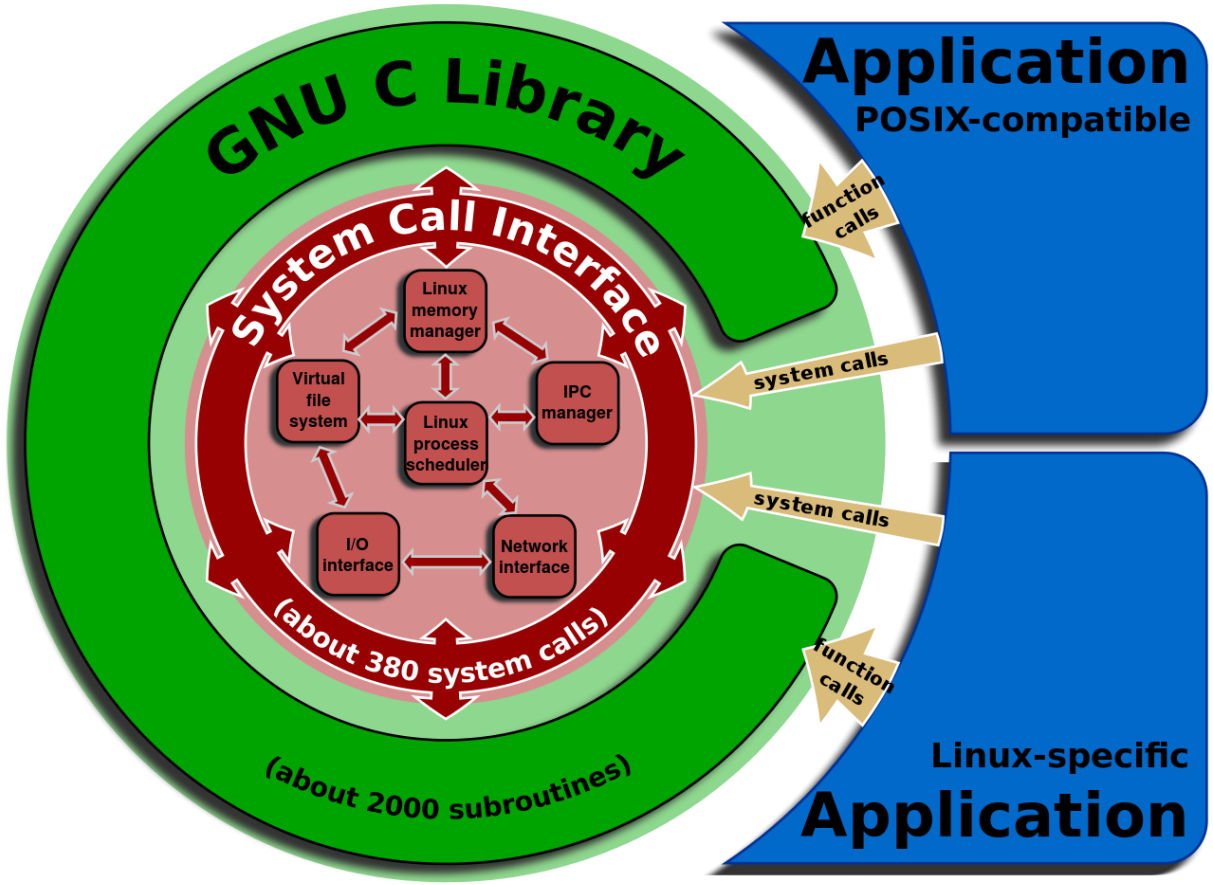
Linux API

- **1.Linux API**
- Linux API是Linux内核与用户空间的API，用户空间的程序能够通过这个接口，访问系统资源和内核提供的服务。Linux API由两部分组成：Linux内核的系统调用接口和GNU C库(glibc)中的例程。

- **2.Linux内核系统调用接口SCI**
- 系统调用接口是内核中，所有已实现和可用系统调用的集合，这是本章重点要介绍的，

- **3.C标准库**
- GNU C 库是Linux内核系统调用接口的封装，下面进一步介绍。Linux内核系统调用接口和glibc库合在一起，就构成了Linux API

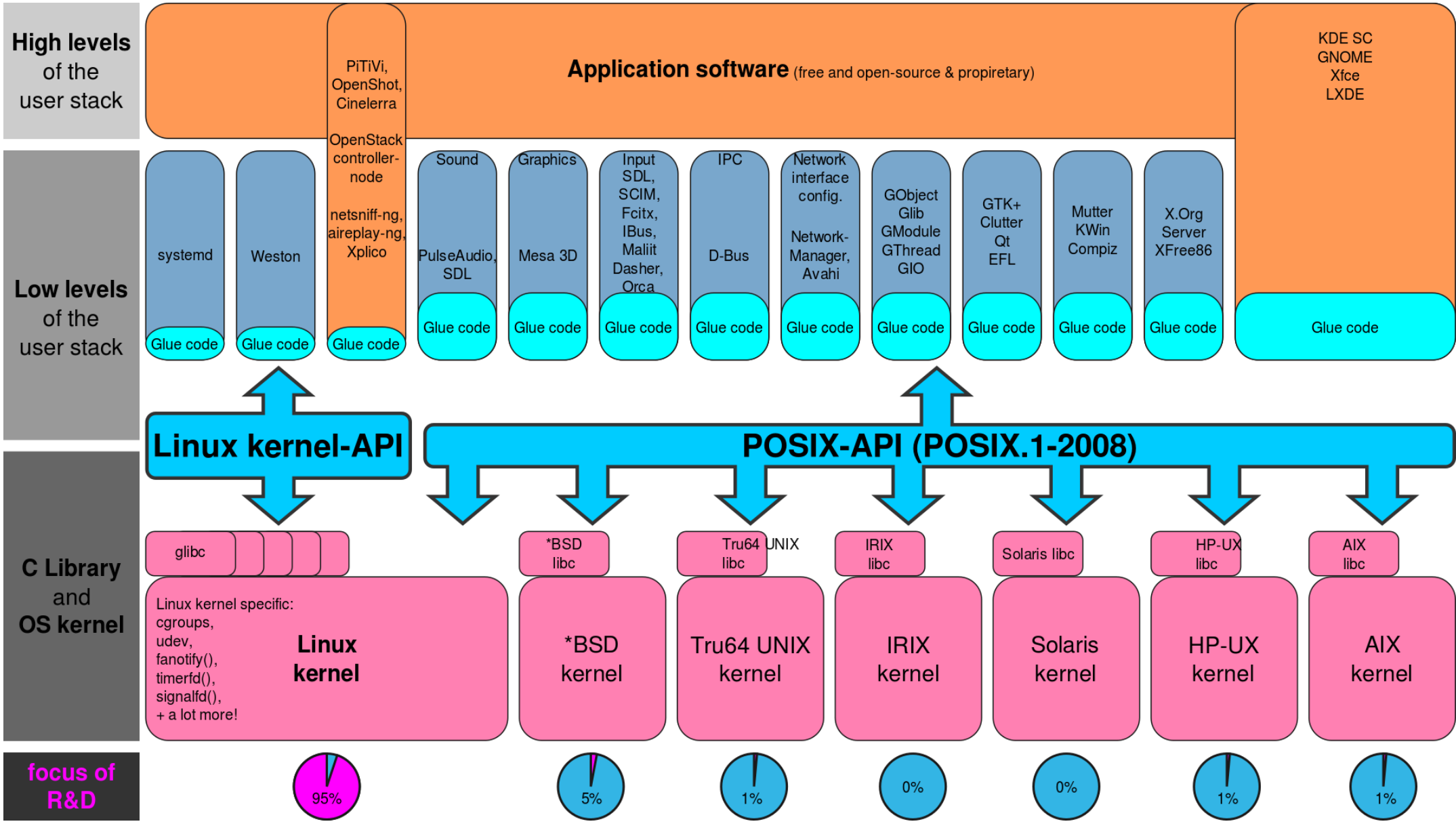
GNU C 库



- GNU C 库是Linux内核系统调用接口的封装。
- 其中包括POSIX兼容应用函数调用，和Linux专用应用的函数调用
- 目前最新Linux内核5.0系统调用有大约有380个左右，GNU C库大约有2000个左右的函数

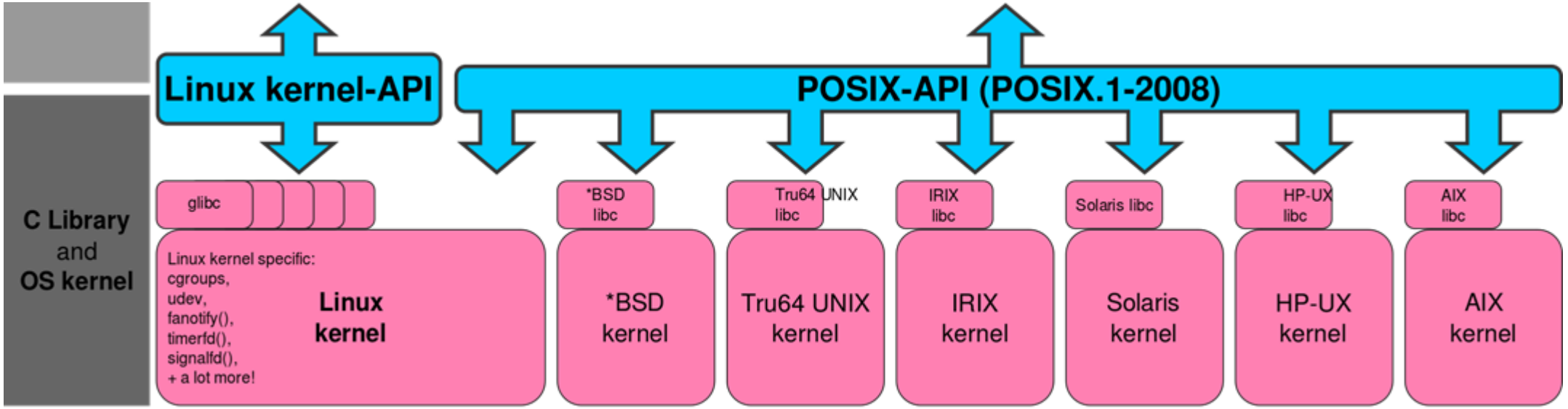
Linux API VS POSIX API

- 如图所示，从下往上。
- 1. 最下面(focus of R&D)是研发人员的比重，
- Linux内核占比最高，95%，
- 其他占比就很小了；

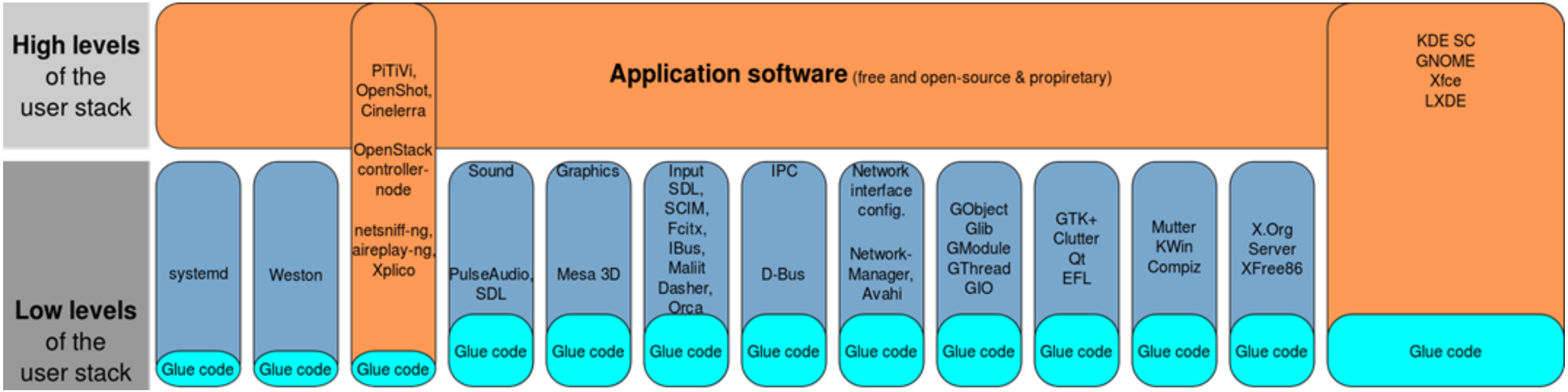


Linux API VS POSIX API

- 2. 第二层是C库和OS内核(C Library and OS kernel)，不同的内核都遵循POSIX的接口，但是就像Linux内核中cgroups，udev这些就不属于POSIX的功能，
- 还可看到，POSIX-API和Linux kernel-API连接着Linux Kernel并向上提供服务；并且就glibc而言，glibc把Linux系统调用，封装进相应API中提供给上层；



- 3. 第三层是用户栈底层(Low levels of the user stack)，像systemd，IPC,QT都不是直接的应用程序，它们依靠胶水函数(Glue code)把上一层提供的API，整合成平台工具供用户使用；
- 4. 最上层是用户栈高层(High levels of the user stack)就到应用软件层了，这一层可以直接调用Linux 内核的API和POSIX-API完成，也可以依托用户栈底层的这些平台工具来构建。

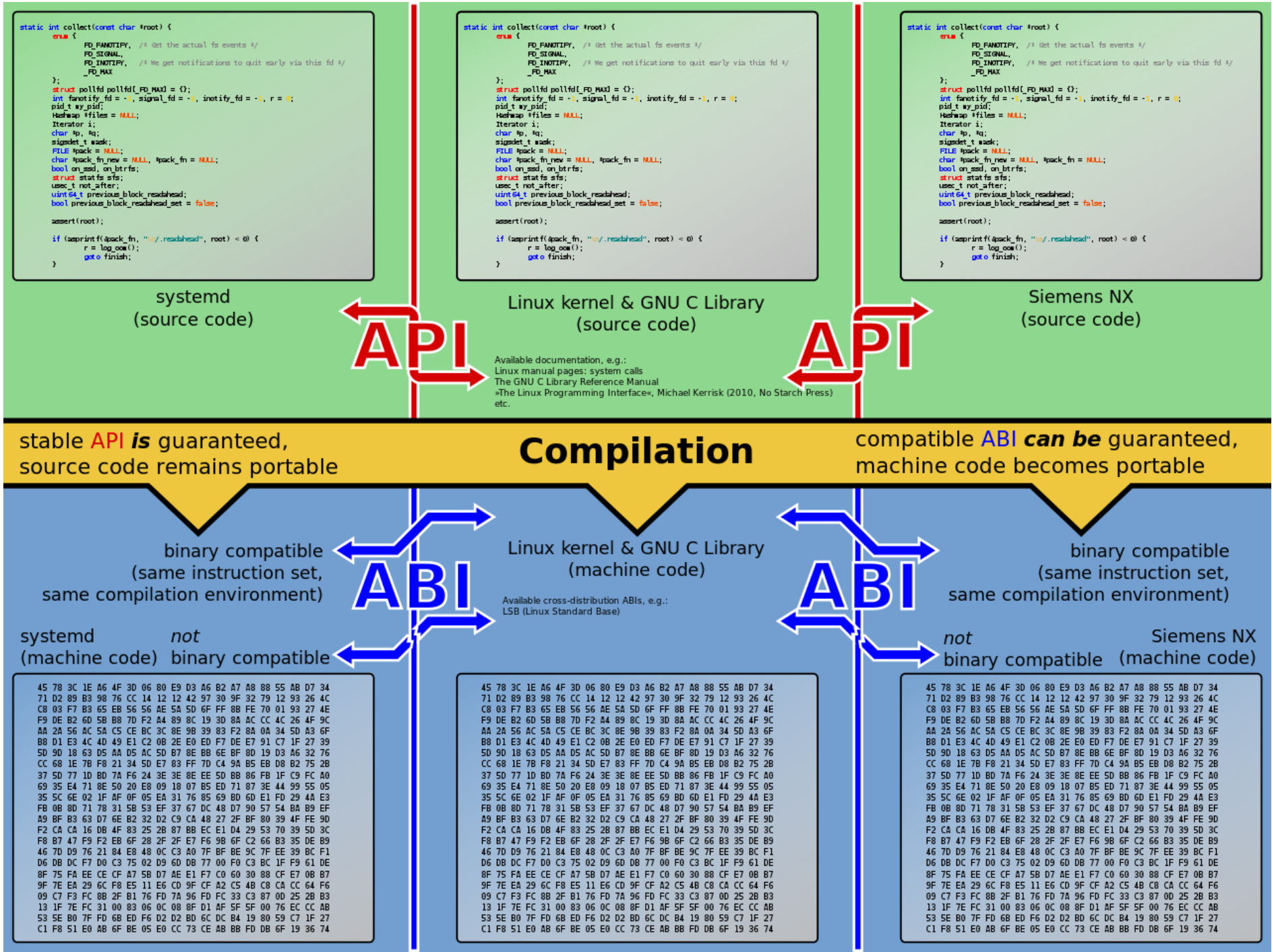


POSIX标准

- 这里对POSIX做一个简介。
- POSIX表示**可移植操作系统接口**(Portable Operating System Interface of UNIX，缩写为POSIX)，POSIX标准定义了操作系统应该为应用程序提供的接口标准，是IEEE为要在各种UNIX操作系统上运行的软件而定义的，一系列API标准的总称，其正式称呼为IEEE 1003。

Linux ABI

- 如图所示。
- 应用程序二进制接口ABI是一系列约定的集合，
- 可以说调用惯例(calling convention)就是ABI。



Linux ABI

- Linux ABI(Application Binary Interface) 是和具体CPU架构和OS相关的。
- 具体而言，ABI包含以下内容：
- 1. 一个特定的处理器指令集
- 2. 函数调用惯例
- 3. 系统调用方式
- 4. 可执行文件的格式(ELF,PE)
- 我们为什么要纠结于ABI这个概念呢？
- 这是为了兼容，只要OS遵守相同的ABI规范，那么不同的应用就可以实现向前兼容， 不用担心版本升级后，旧版本的应用不能运行了。

内核API

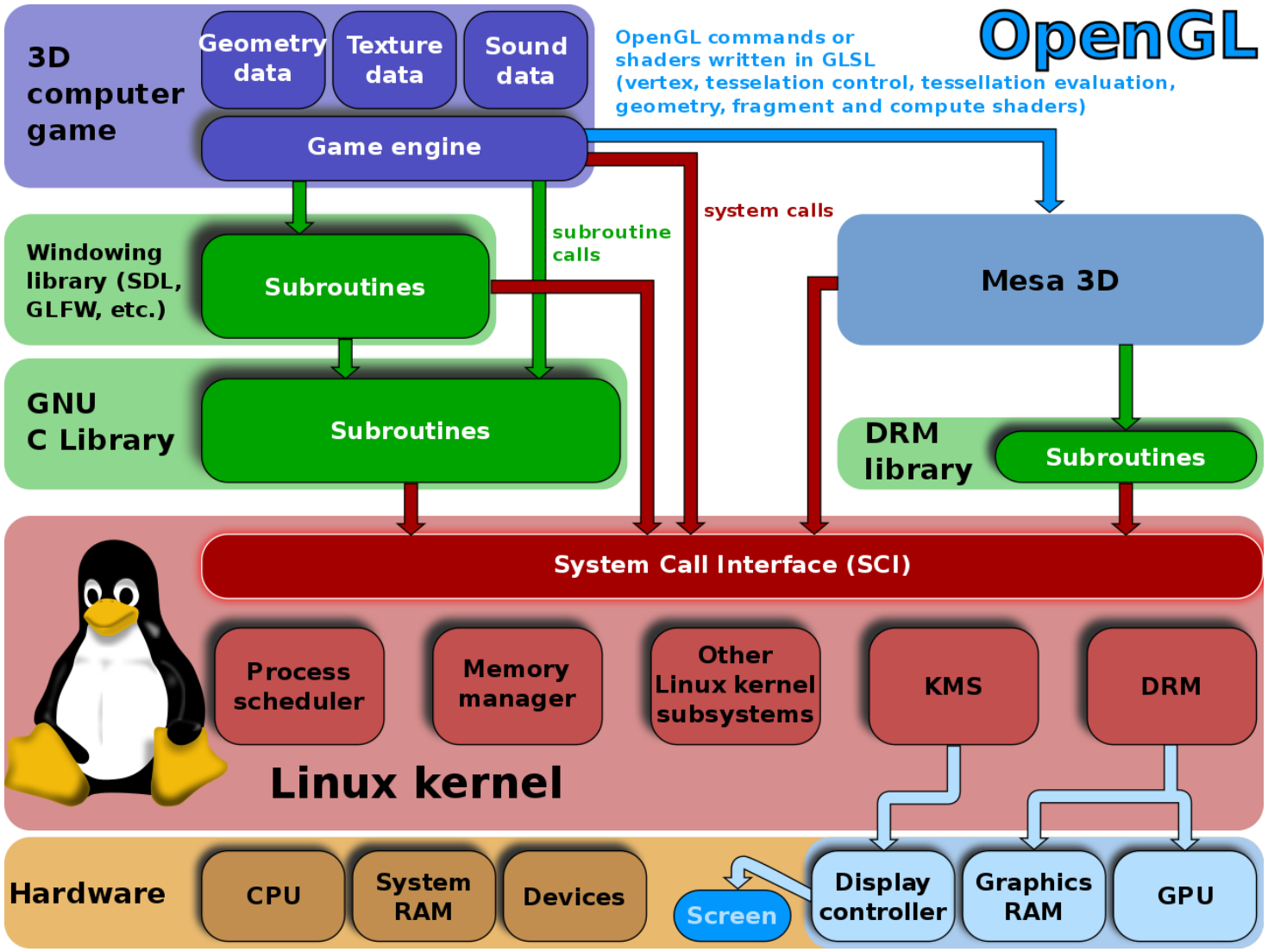
```
void add_wait_queue(wait_queue_head_t *q,
wait_queue_t *wait)
{
  unsigned long flags;
  wait->flags &= ~WQ_FLAG_EXCLUSIVE;
  spin_lock_irqsave(&q->lock, flags);
  __add_wait_queue(q, wait);
  spin_unlock_irqrestore(&q->lock, flags);
}
EXPORT_SYMBOL(add_wait_queue)
```

在kernel/wait.c

- 内核API主要是内核中标记为“EXPORT_SYMBOL”的函数。
- 这些函数主要是为了内核模块的编写而提供的。
- 受到内核版本迭代的影响，内核API并不稳定。3.x版本内核的模块可能在4.x版本上就无法使用。

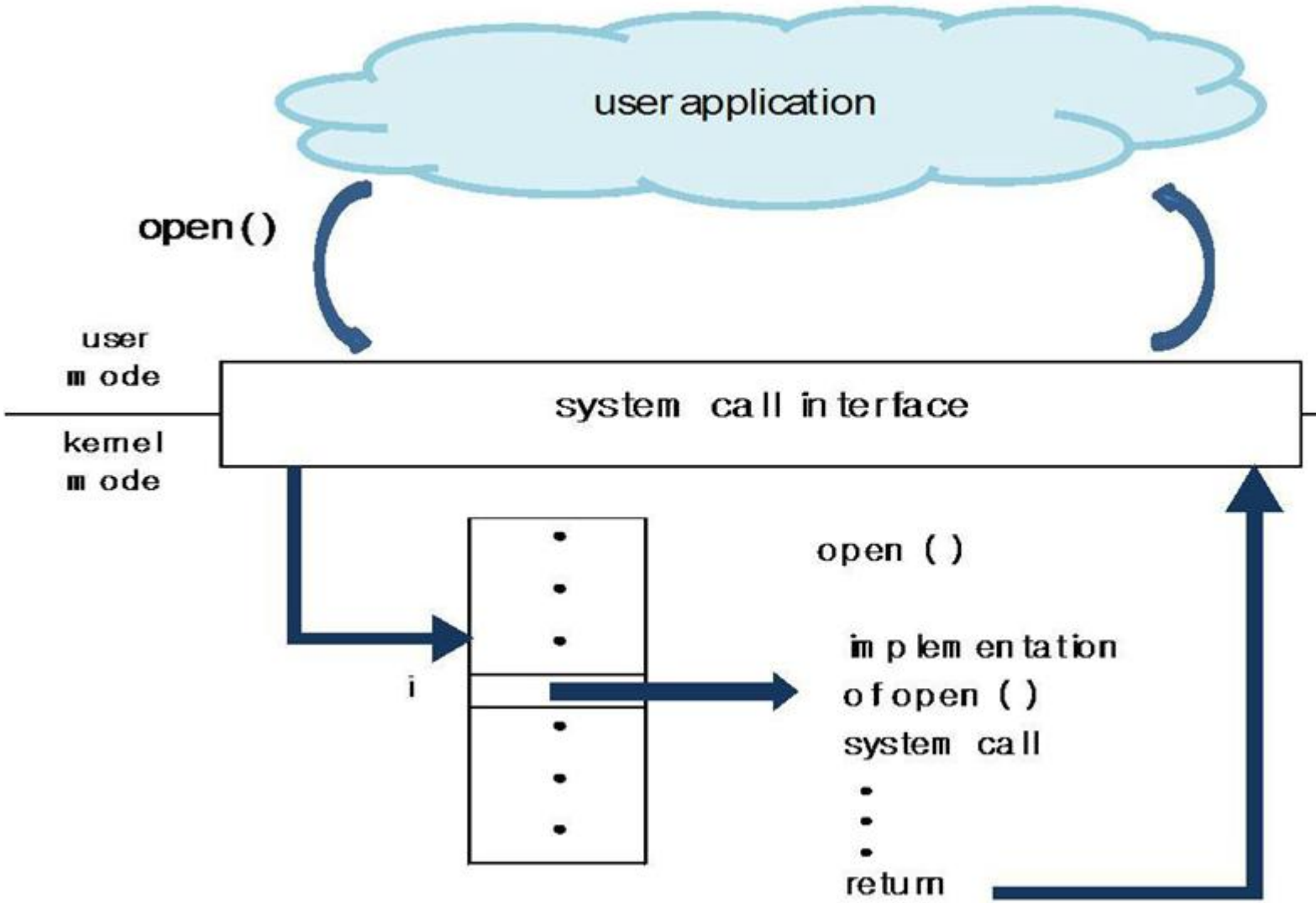
抽象API

- 在某些情况下，内核过于底层，开发者需要更高一层的抽象。
- 于是出现了类似Mesa 3D的为图形驱动开发而生的API。
- 如图所示。



2 系统调用机制

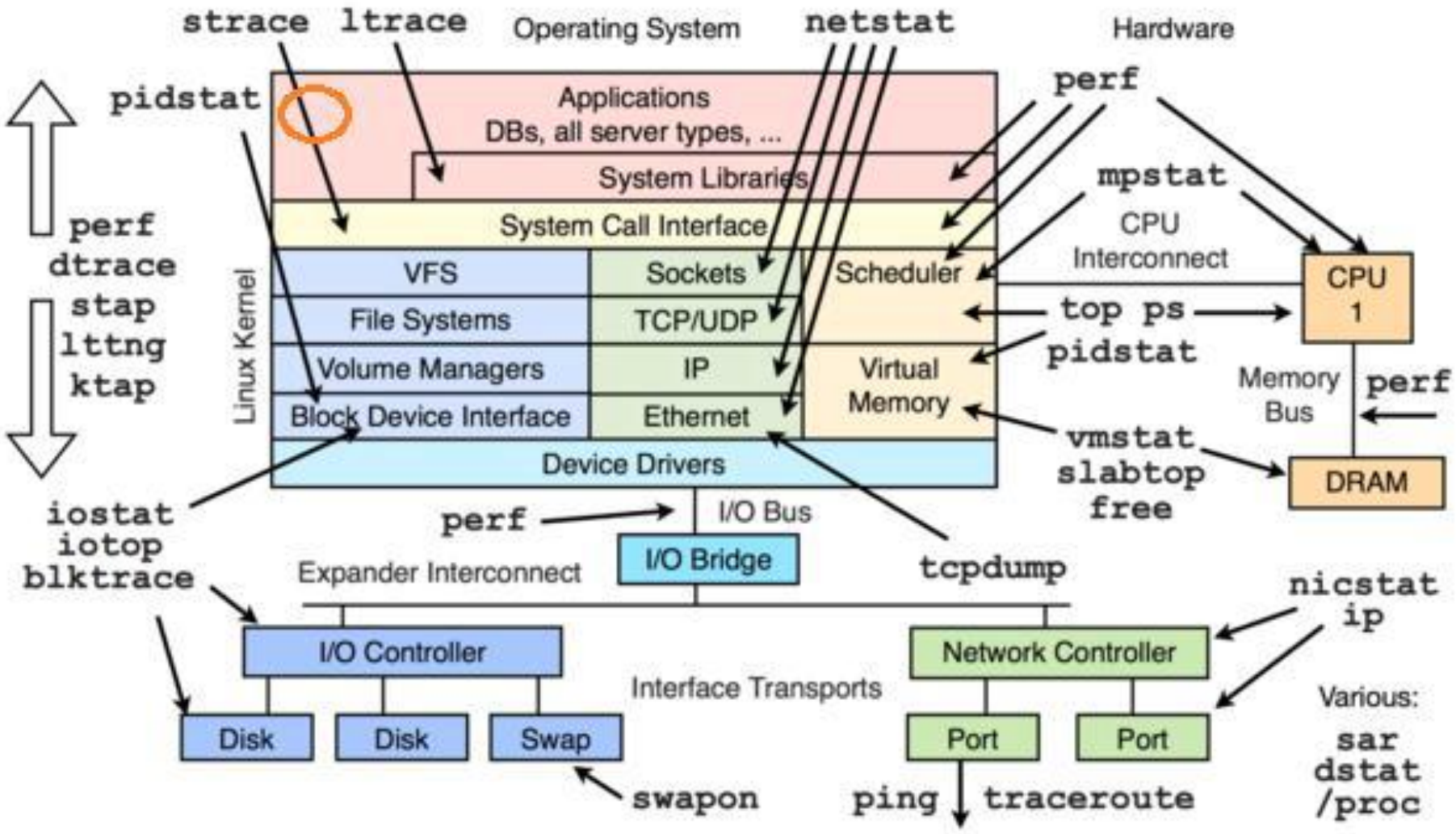
系统调用 - 内核的出口



- 系统调用，顾名思义，说的是操作系统提供给用户程序调用的一组“特殊”接口。
- 从逻辑上来说，系统调用可被看成是一个内核与用户空间程序交互的接口
- 它好比一个中间人，把用户进程的请求传达给内核，待内核把请求处理完毕后，再将处理结果送回给用户空间。
- 如图所示。

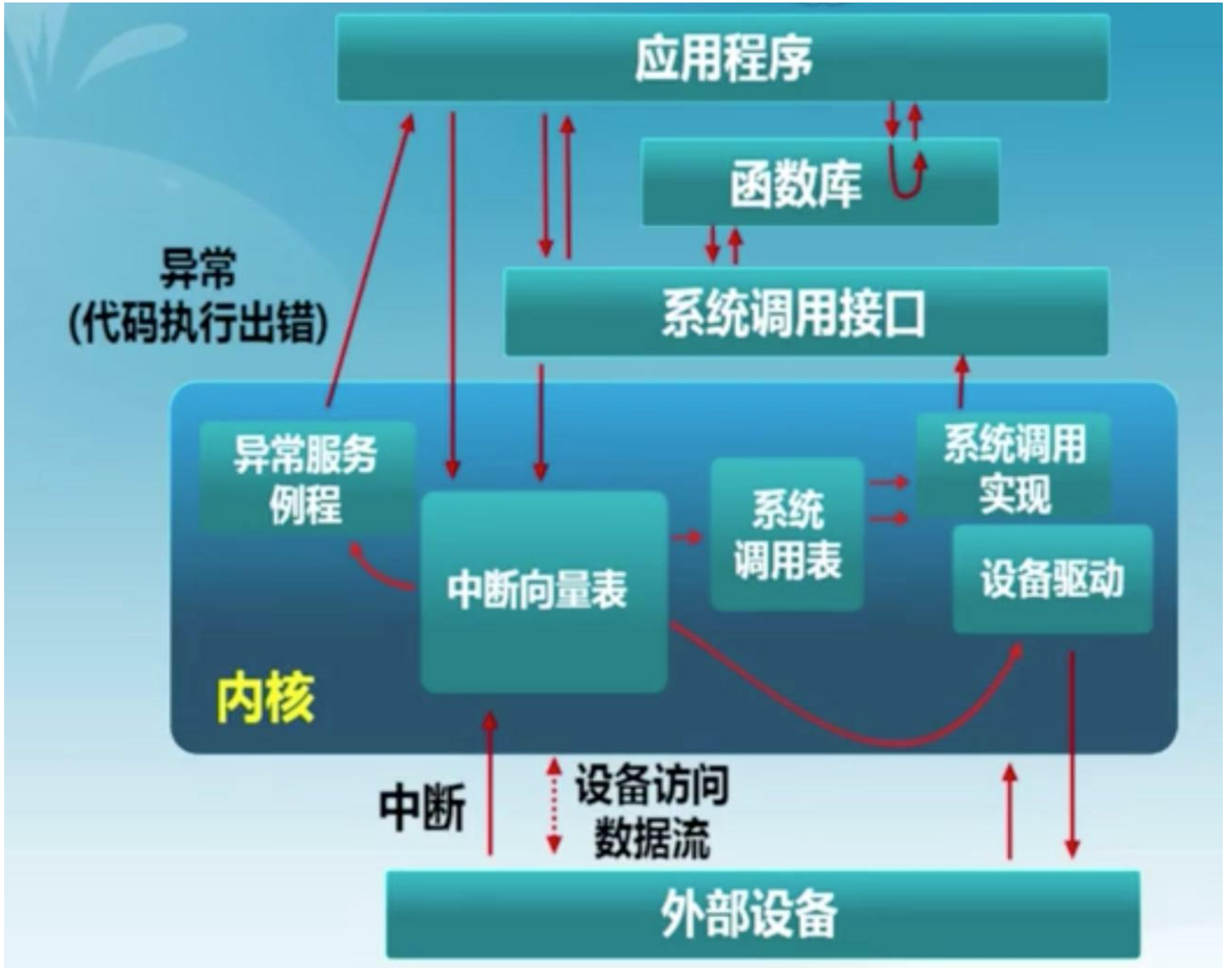
跟踪进程所调用的系统调用

- 如图所示为Linux系统中各个子系统相关的工具。
- 可以通过strace命令查看一个应用所调用的系统调用
- 比如$strace ls



中断、异常和系统调用比较

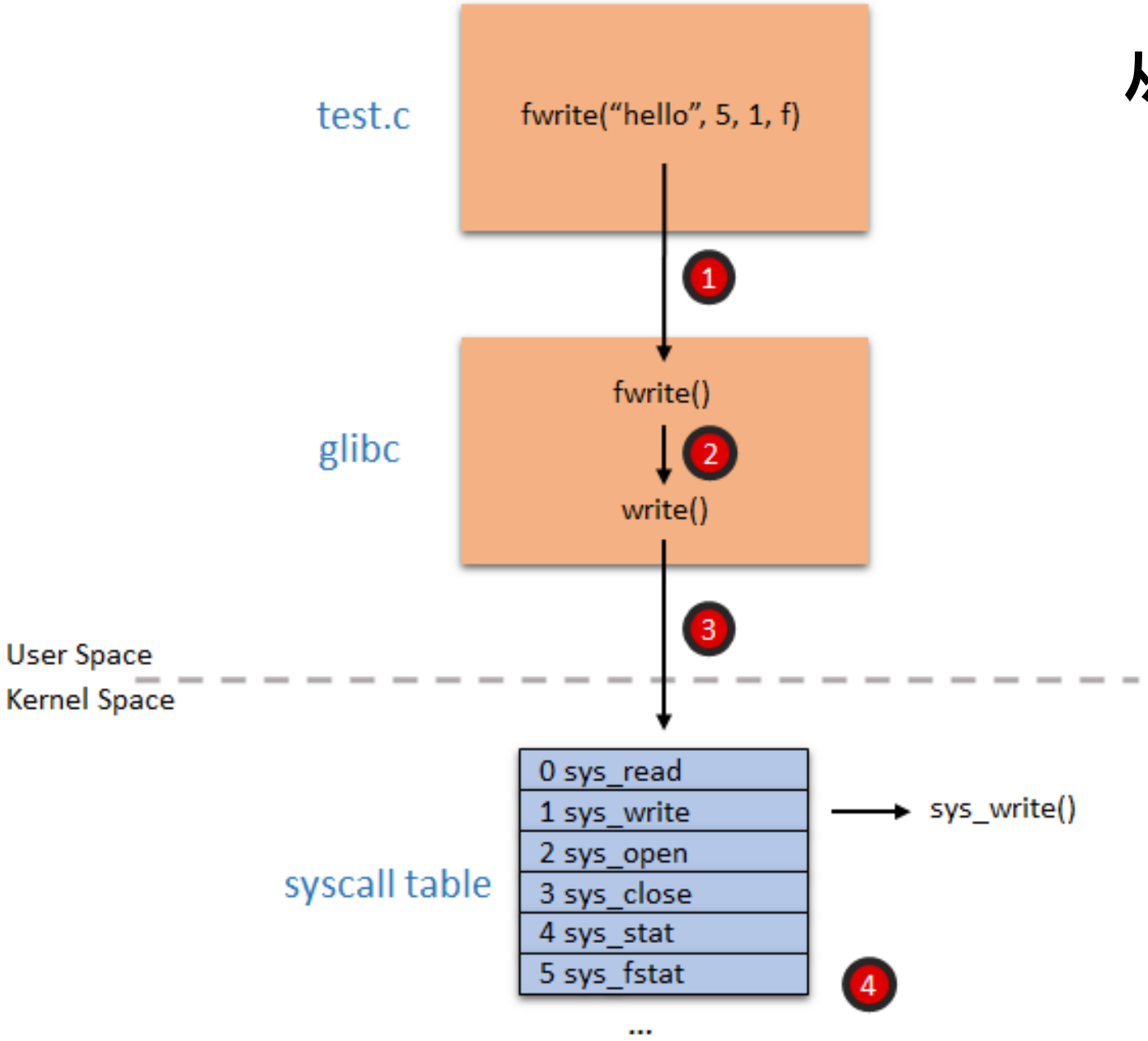
- 中断、异常和系统调用本质是属于一类，处理方式上也类似，
- 那么它们之间的差异性表现在哪些方面，
- 在学习了中断后，对学系统调用有哪些帮助？



中断、异常和系统调用的三个不同

- 1)源头不同
  - 中断:是外设发出的请求
  - 异常:是应用程序意想不到的行为
  - 系统调用：应用程序请求OS提供
- 2)服务响应方式不同：
  - 中断是异步的，异常是同步的，而系统调用既可以是异步，也可以是同步。
- 3)处理机制不同
  - 中断服务程序，在内核态下运行，对用户是透明的。
  - 异常出现时，或者杀死进程，或者重新执行引起异常的指令。
  - 系统调用，用户发出请求后等待OS的服务。

从用户态函数到系统调用的例子

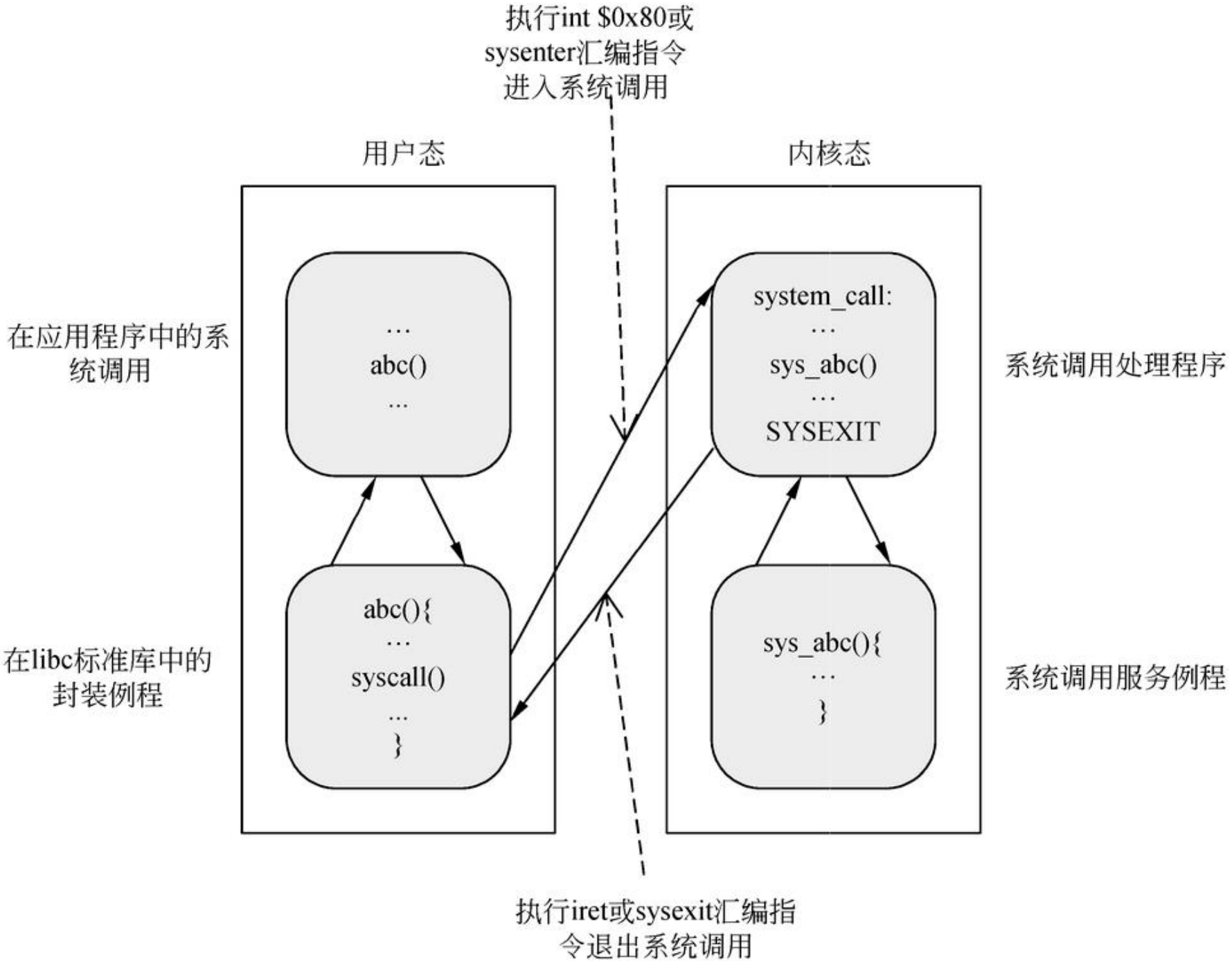


- 比如在程序中调用fwrite函数，而fwrite函数在glibc库中调用系统调用write(),
- 然后从用户态陷入内核态，查找系统调用表，对应的系统调用服务例程为sys_write

3 系统调用流程

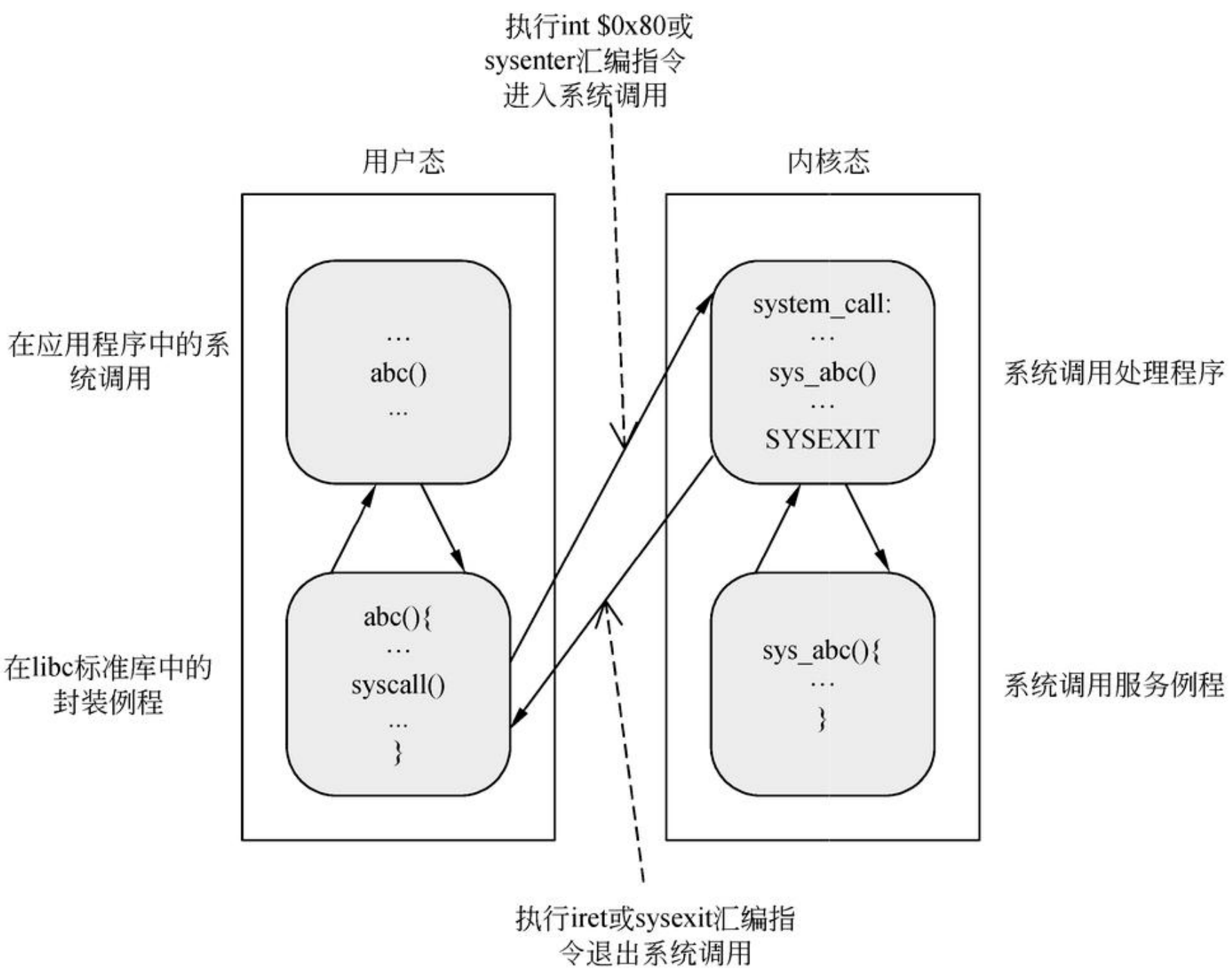
系统调用一般处理流程

- 如图所示。
- 当用户态的进程调用一个系统调用时，在libc的封装例程中，会调用int0x80或者syscall汇编指令，切换到内核态，
- 并开始执行一个内核system_call系统调用处理程序



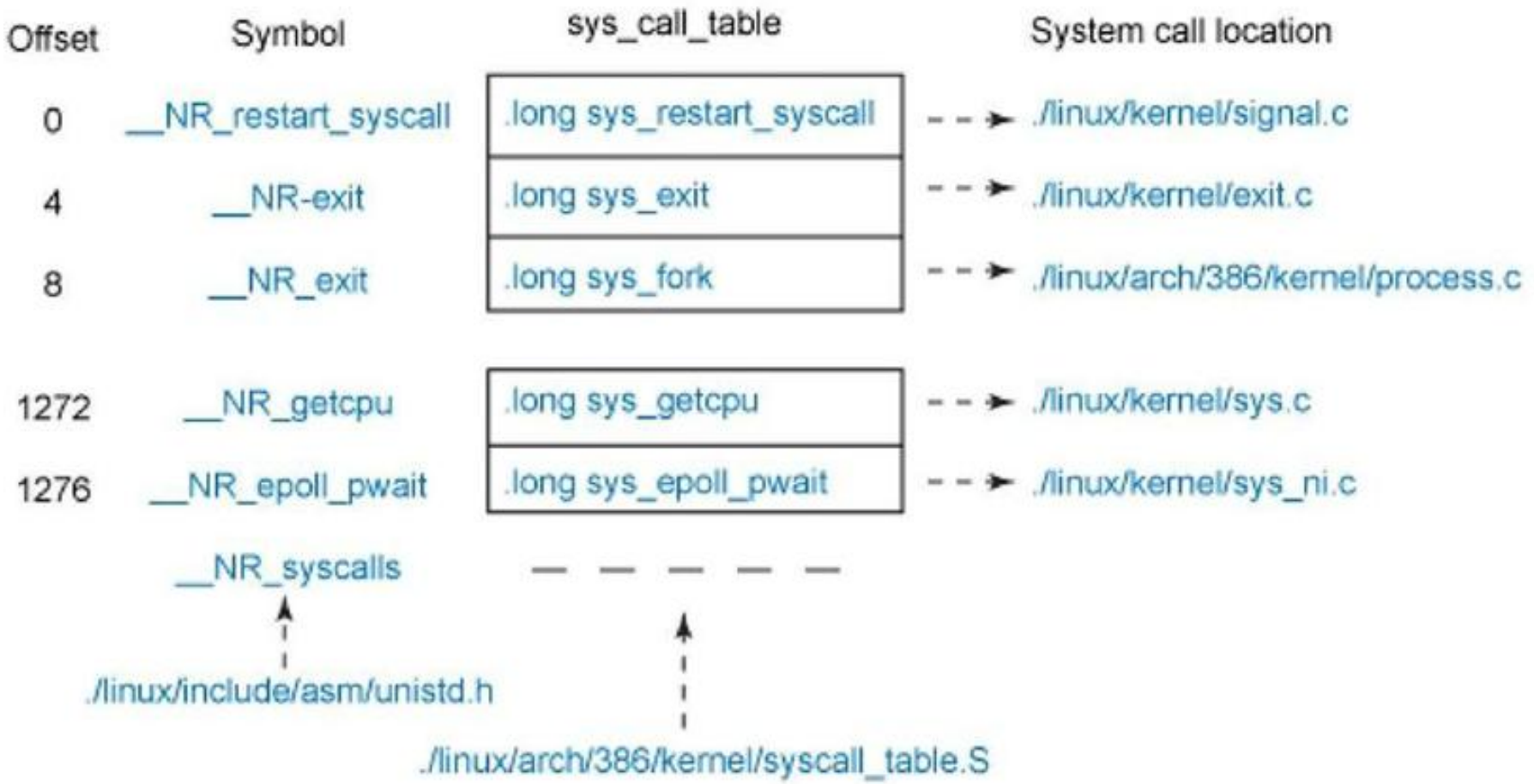
系统调用一般处理流程

- 系统调用处理程序执行下列操作：
- 在内核栈，保存大多数寄存器的内容
- 调用系统调用服务例程，来处理系统
- 调用通过iret或者sysexit汇编指令，从系统调用返回



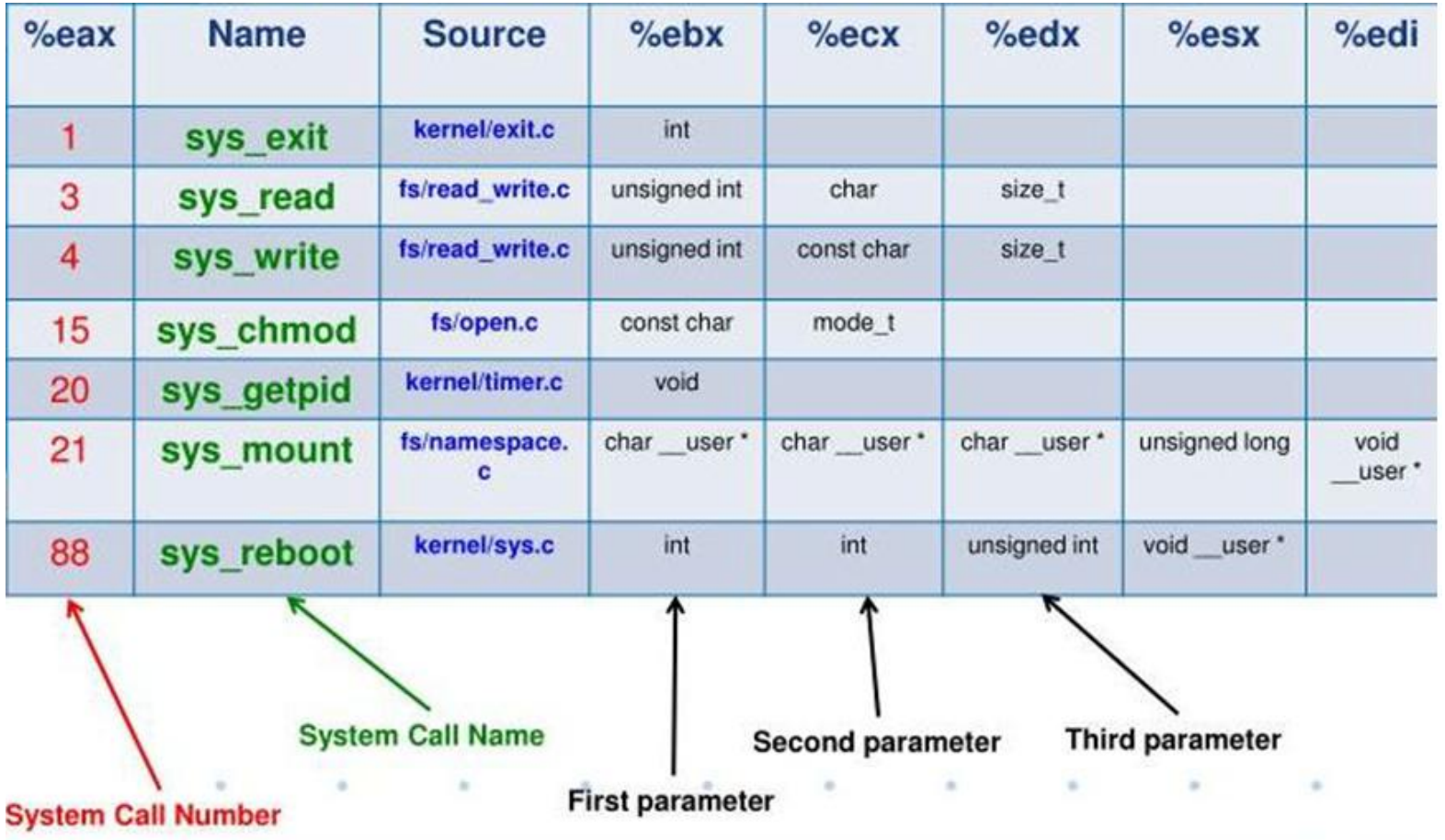
系统调用基本概念

- 系统调用号 作用是：
- ①用来唯一的标识每个系统调用；
- ②作为系统调用表的下标，当用户空间的进程执行一个系统调用时，该系统调用号，就被用来指明，到底要执行哪个系统调用服务例程。



系统调用列表

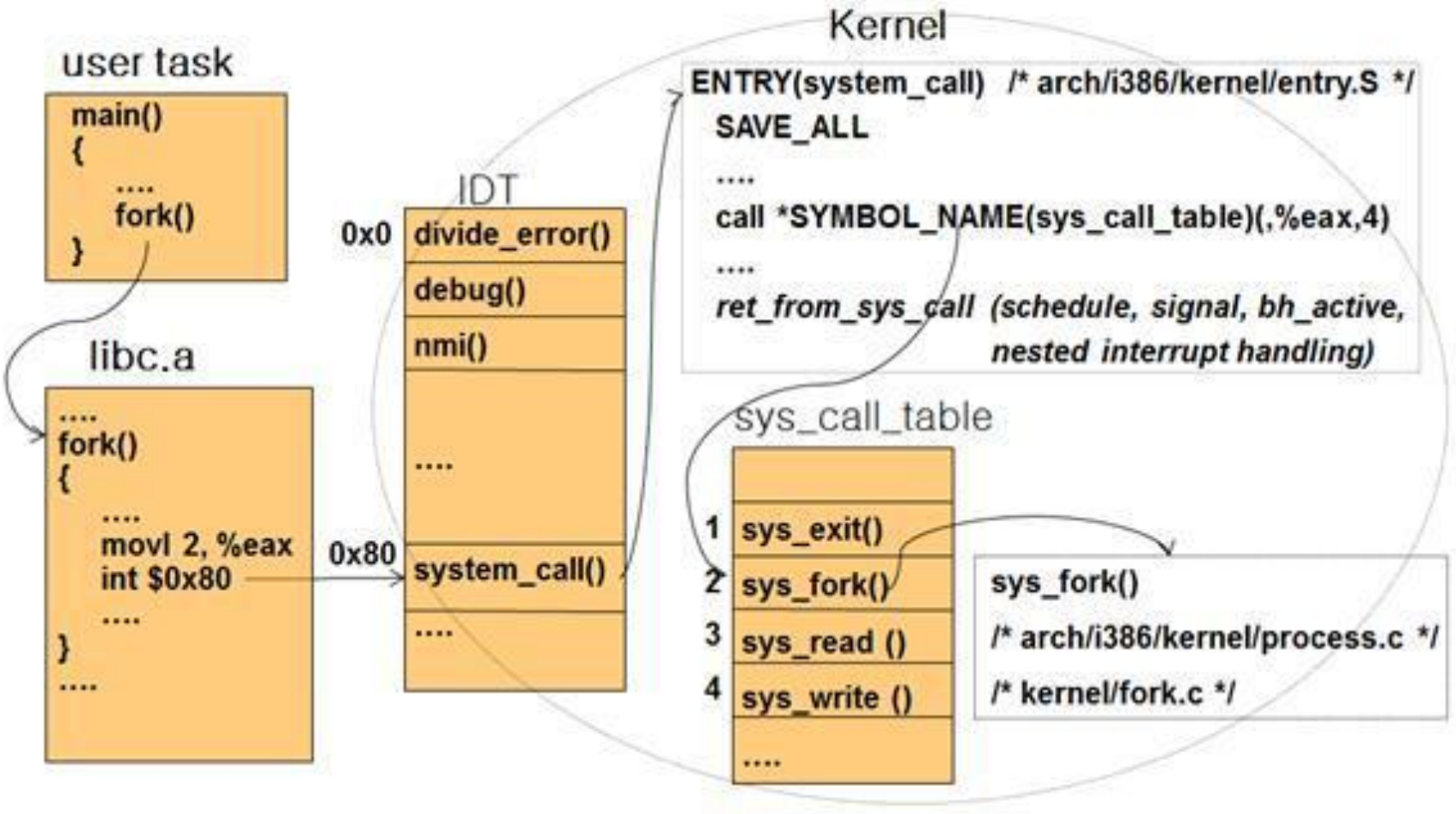
- 系统调用表 作用是：
- 将系统调用号和相应的服务例程关联起来，该表存放在sys_call_table数组中：
  - 内核版本不同，系统调用号，和系统调用的头文件会有差别
- 从表看出，系统调用号存放在eax寄存器中。
- 每个系统调用在内核中对应的服务例程以sys打头,
- 它们的实现所在的源文件也各不相同，系统调用的参数存放在寄存器中，一般参数不超过6个(包括系统调用号)。



%eax	Name	Source	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	kernel/exit.c	int				
3	sys_read	fs/read_write.c	unsigned int	char	size_t		
4	sys_write	fs/read_write.c	unsigned int	const char	size_t		
15	sys_chmod	fs/open.c	const char	mode_t			
20	sys_getpid	kernel/timer.c	void				
21	sys_mount	fs/namespace.c	char __user *	char __user *	char __user *	unsigned long	void __user *
88	sys_reboot	kernel/sys.c	int	int	unsigned int	void __user *	

从用户态跟踪一个系统调用到内核

- 1.从用户程序中调用fork
- 2.在libc库中把fork对应的系统调用号2放入寄存器eax
- 3.通过int 0x80陷入内核
- 4.在中断描述表IDT中查到系统调用的入口0x80
- 5.进入Linux内核的entry_32(64).S文件，从系统调用表sys_call_table中找到sys_fork的入口地址
- 6.执行fork.c中的do_fork代码
- 7 通过iret或者sysiret返回
- 如图所示。



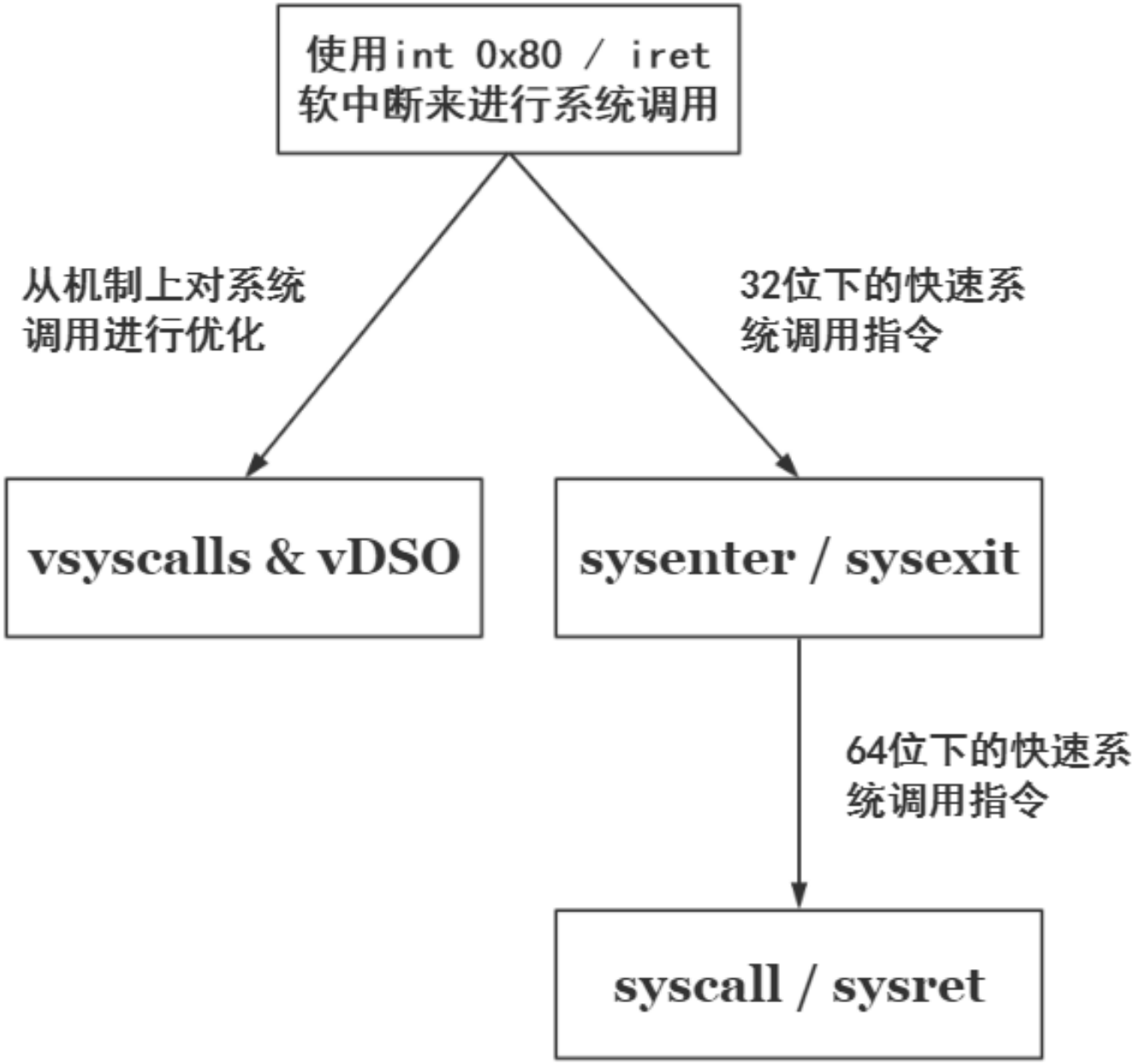
4 系统调用的优化

系统调用机制的优化

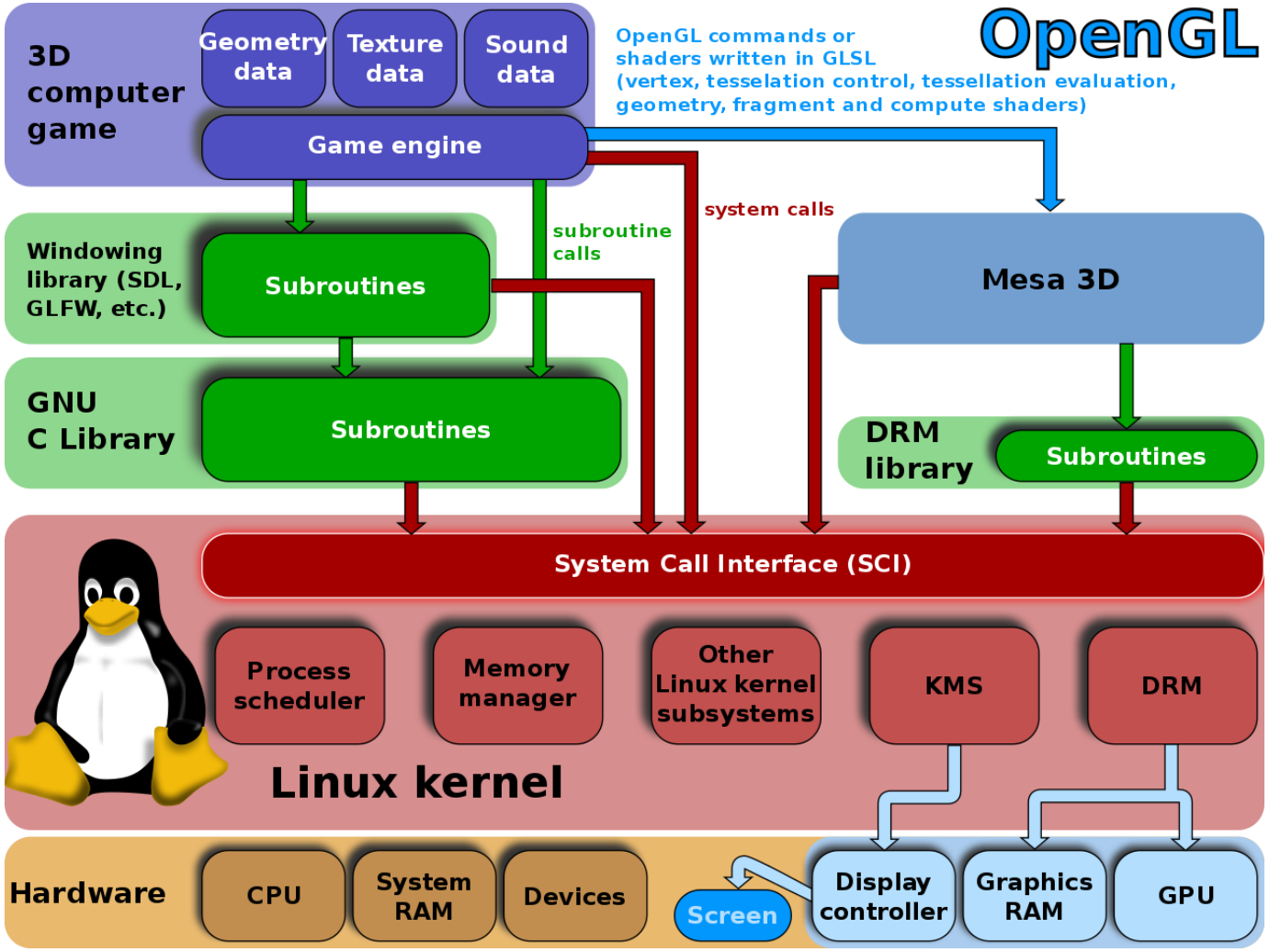
- 在2.6的早前版本中，系统调用的实现用的是int0x80和iret命令。
- 因为系统调用的实现从用户态切换到内核态，执行完系统调用程序后又从内核态切换回用户态,代价很大，
- 为了加快系统调用的速度，随后先后引入了两种机制——vsycalls和vDSO 。
- 这两种机制都是从机制上，对系统调用速度进行的优化，但是使用软中断来进行系统调用，需要进行特权级的切换，这一根本问题没有解决。

系统调用机制的优化

- 为了解决这一问题，Intel x86 CPU从Pentium II之后，开始支持快速系统调用指令sysenter/sysexit，
- 这两条指令是Intel在32位下提出的，而AMD提出syscall/sysret，64位统一使用这两条指令了。
- 如图所示。



系统调用小结



- 系统调用是应用与内核之间的一个接口
- Linux内核中系统调用的具体实现与CPU体系结构相关。
- 是否要添加一个新的系统调用，需要认真评估。

课程思政

课程思政

提高职业素养，助力职业成长，早日实现人生价值

早在20世纪70年代，国内许多科研院所就已参与到了UNIX自主操作系统的研发中。伴随着Linux系统的诞生，其凭借开源特征得以迅速取代UNIX成为国产操作系统开发的主流。

20世纪90年代，曾是中国Linux公社（Linuxfans）的发起人和重要参与者之一的黄建忠，做了一个彻底改变他人生轨迹的选择——投身北京中科红旗软件技术有限公司（简称中科红旗）。他带领团队以Linux系统为基础二次开发操作系统，经过多年的努力，国产操作系统无论是布局还是操作方式，都同Windows XP相差无几，并在易用性等方面基本具备了Windows XP替代能力。然而，当黄建忠真正走进市场才发现，原本以为是“蓝海”的操作系统领域，其实是“死海”！

提高职业素养，助力职业成长，早日实现人生价值

2009年，中国电子信息产业的国家队——中国电子科技集团，整合集团优势资源，投资设立普华公司，黄建忠又一次义无反顾地投身其中，成为普华技术部研发总监，中科红旗的创业团队也先后投奔过来。普华肩负提升国家基础软件产业核心竞争力的重要使命，2014年初正式进军国产操作系统领域。

当年9月普华操作系统3.0版本便正式发布。但是走进市场才发现，绝大多数人依旧只认Windows！直到“棱镜门”事件爆发与WannaCry（音译“想哭”）病毒肆虐全球，网络安全才逐步上升到国家战略层面，具备网络安全优势的国产操作系统逐渐深入人心，得到了业内外的普遍认同，并逐步进入国家政府部门以及金融、能源等经济社会运行的神经中枢。

我们作为未来的科技工作者，要向黄建忠学习，提高职业素养，助力职业成长，早日实现人生价值。

课后练习题

1、Linux中有四种类型的接口，请对其进行简述。
2、请对POSIX（Portable Operating System Interface of UNIX）标准做一个简要介绍。
3、请简述系统调用（system call）的含义及其主要功能。
4、请简述中断、异常和系统调用的区别。
5、系统调用一般处理流程是什么，请简述。
6、从用户态跟踪一个系统调用到内核有哪些步骤，请简述。
7、大部分的应用程序运行在什么空间，内核和设备驱动运行在什么空间，请简述。

谢 谢！

THANKS