**POSIX API和系统调用：**

  系统调用是操作系统为用户态的进程和硬件设备进行交互提供的一组接口

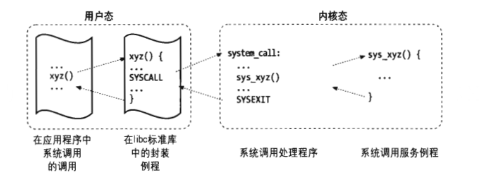
  API接口和系统调用的不同：前者只是一个函数定义，说明了如何获得给定的服务，后者是通过软中断向内核发送了一个明确

的请求，系统调用属于内核，但是用户态的库函数不属于内核。

  API可能直接提供用户态的服务，也可能调用了几个系统调用。此外，几个API也可能调用封装了不同功能的同一系统调用，例如，

libc库中的malloc()，calloc()，free()都是通过brk()系统调用来实现。

**系统调用处理程序与系统调用服务例程：**



  用户程序通过int 0x80软中断陷入内核态，int 0x80软中断的调用一般交给库来封装了，比如上图的xyz()方法，一般都是调用库而

不是调用0x80这样的数字进入。通过软中断 0x80,系统会跳转到一个预设的内核空间地址,它指向了系统调用处理程序。

  system\_call(系统调用处理程序，或名为系统调用例程)， 是用汇编代码定义的，通过这个函数，可以从用户态转换到内核态，然

后在内核态中通过“调用号”调用不同的“服务例程”。其实服务例程才是实际上处理数据的程序。而系统调用处理程序，只是实现了

从用户态到内核态转换后的一些必要处理而已。

  所有的系统调用都会统一跳转到这个地址进而执行 system\_call 函数,但到2.6.23 版为止,内核提供的系统调用已经达到了 325 个，

因此软中断指令 int 0x80 执行时会有一个系统调用号被会被放入 eax 寄存器，这样system\_call 函数可以读取 eax 寄存器获得当系

统调用的系统调用号，将其乘以 4 生成偏移地址，加上 sys\_call\_table 的基址指向的内容即是应该执行的系统调用服务例程的地址。

sys\_call\_table是一个把系统调用号和服务例程关联起来的一个表。

**进入系统调用和退出系统调用：**

    进入：

* 执行 int $0x80指令
* 执行sysenter指令，linux2.6以后支持该命令

       退出：

* 执行iret指令
* 执行sysexit指令

  通过system\_call函数进入系统调用：

    首先把系统调用号和相应的内容存入对应的栈中，然后对用户态传进来的系统调用好进行有效性检查，如果非法则终止系统调用。

系统调用结束会有一个返回码，system\_call函数会把这个返回码放在存放用户态eax寄存器内容的那个栈单元里，这样进程返回用户态执行

时就会发现一个返回码。

  通过sysenter指令进入系统调用(相对于int指令速度更快)：

    封装例程把系统调用号放入eax寄存器，然后调用\_\_kernel\_vsyscall()函数把寄存器的值保存到用户堆栈中，把用户栈指针存入ebp中，执

行sysenter指令，CPU从用户态切到内核态，内核执行sysenter\_entry()函数把用户栈指针，用户态代码段段选择符地址，系统调用结束要执、

行的指令地址存入到内核态堆栈中，通过一系列指令调用系统调用处理程序。

    sysenter用到了几个特殊寄存器，在执行sysenter指令时，cpu控制单元把SYSENTER\_CS\_MSR(内核代码段选择符)，

SYSENTER\_EIP\_MSR(内核入口线性地址)，SYSENTER\_ESP\_MSR(内核堆栈指针)，SYSENTER\_CS\_MSR+8（内核堆栈段选择符）分别

拷贝到cs,eip,esp,ss寄存器中。系统调用执行完毕后，这些寄存器的值都被恢复为用户态的。

**参数传递：**

  系统调用也需要传入或传出参数，这些参数可能是实际的值，也可能是地址。system\_call和system\_entry()是所有系统调用的公共入口，因此

至少有一个系统调用号参数。一般函数传递参数是把参数值写入程序栈（用户态或者是内核态），但是系统调用是跨用户和内核态的特殊函数，

所以只能通过寄存器的方式来传递参数。在调用系统调用服务例程前，把参数写入CPU寄存器，内核再把寄存器的值拷贝到内核态的栈中。

因此每个参数长度不能超过寄存器长度即32位。当参数长度超过六个时，寄存器的数量不足，这时就会用一个寄存器指向一个存放这些参数的

内存区。

**内核封装例程：**

一般系统调用只有用户态进程使用，但有时也被内核线程所使用，但内核线程不能直接使用库函数。linux定义了\_syscall110~\_syscall116这7个

宏，0~6代表系统调用传递的参数个数，每个宏需要2+2\*n个参数，前两个参数是系统调用的类型和名称，之后每一对指明参与系统调用的

参数的类型和名称，如下

  内核态调用fork()：\_\_syscall110(int , fork)

  内核态调用write()：\_\_syscall113(int , write, int, fd, const char \*, buf, unsigned int, count )