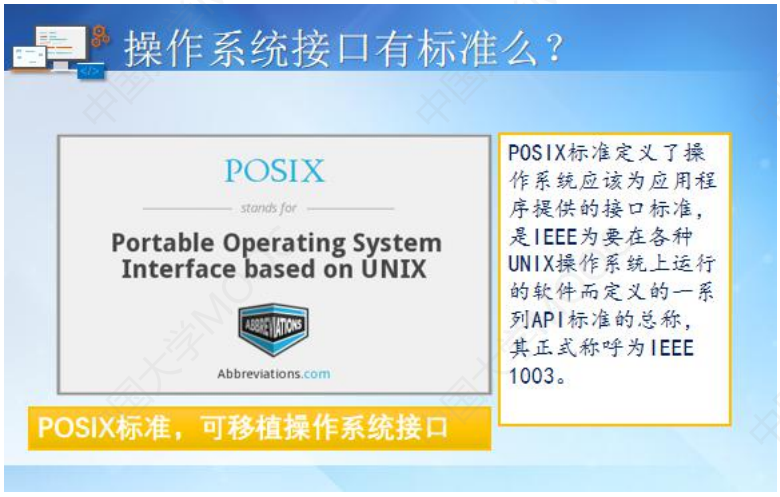
操作系统接口课前预习宝典

一、带着问题上路

什么是操作系统？从第一章我们了解到，操作系统管理着计算机的软硬件资源，并为计算机程序提供服务，从图可以看出操作系统是应用软件和硬件之间的桥梁，为什么说 OS 是桥梁呢，我们会逐渐了解到，操作系统的对硬件的管理，最终以服务的形式提供给用户，比如，从磁盘读数据，因为与硬盘打交道是件非常麻烦的事，于是就由 OS 的设备管理来做，最后读出来的数据，通过文件系统交给用户。

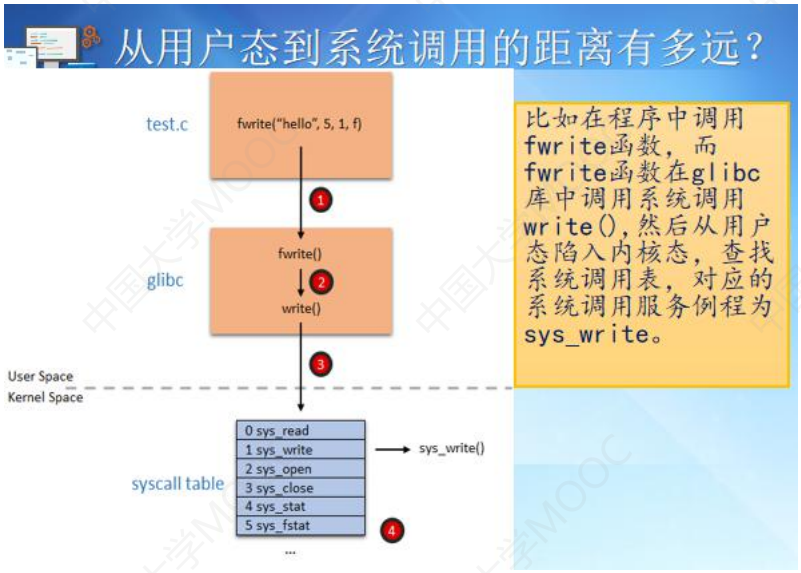
那么 OS 的大管家角色，如何体现，主要体现在对 CPU（也就是进程）的管理，对内存的管理，对设备的管理，以及对文件的管理，服务员的角色又如何体现，依然举前面从磁盘读数据的例子，谁去磁盘读数据，实际上是磁盘的驱动程序干的，把数据读好后就直接交给用户么，没那么简单，先交给文件系统，用户通过文件系统的 read()函数就把数据拿到了，这个函数就是服务员的角色，就像服务员把大厨子（相当于 OS）做好的菜端给你。



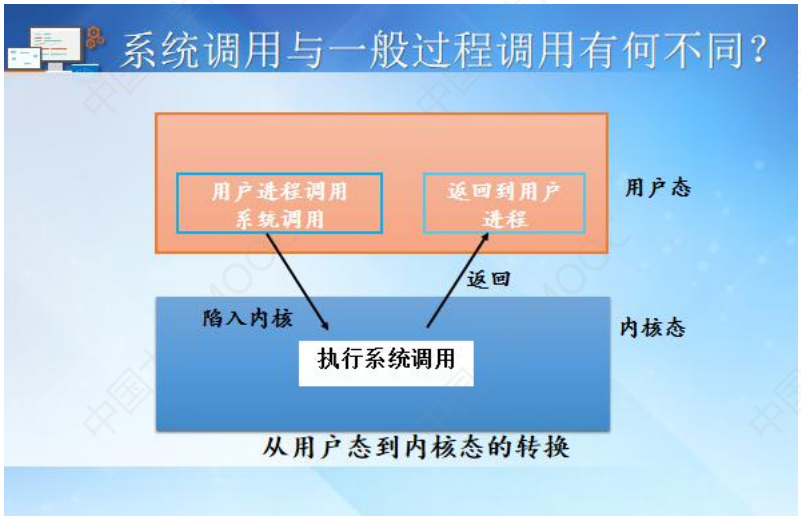
所谓接口就是两个物件相连的部位，对软件来说，就是软件不同部分衔接的一种约定。操作系统有什么接口么，这就是大名鼎鼎的 POSIX 标准。什么是 POSIX 标准，就是 Portable Operating System Interface of UNIX 的缩写，正是它提供了操作系统应该为应用程序提供的接口标准，这样的标准会带来什么好处？好处太大了，只要遵循这一标准，一方面别人家的衣服就可以穿到你的身上了，比如，Unix 上程序就可以在 Linux 上跑，另一方面，标准给出了函数的定义形式，比如，int fork()的形式， 至于函数如何实现，各个操作系统就可以各显其能了，但函数对外提供的接口就是这种形式。

操作系统的接口有哪些类型？这里给出三大类，图形接口，命令级接口和程序接口。
图形接口使得 80 岁的爷爷和三岁小朋友都可以使用计算机，但你想使得计算机做更多的事情，图形接口就显得力不从心了，对于系统管理员或者程序员来说，可以通过命令接口向计算机发号施令，Linux 的普通用户可以使用 200 多条命令，系统管理员还可以使用 200 条命令，每条命令又有很多选项，这是图形界面无法给予的。而对于开发者来说，你实际上是通过函数库享用操作系统的劳动成果的，但是一般来说，你并没有感受，可能还觉得就是调用了一下库函数而已，实际上，库函数后面站着一个大公无私的巨人，这就是操作系统对系统调用的实现。

不管是图形接口还是命令接口，归根结底，都会调用一个接口，这就是系统调用，比如，当我们要从磁盘上读取一个文件时，这是一件非常复杂的事情，这件事就交给 OS 来做，我们用户程序通过系统调用接口向 OS 发出 read()请求，OS 就把它辛辛苦苦读取的结果很乐意的交给我们，从而把我们程序员从与硬件打交道的繁杂事务中解放出来，诸如此类，还有很多的脏活累活我们都可以交给 OS 去做，OS 把完成后的结果以系统调用接口形式呈现出来，让我们很方便的享用它的劳动成果，这就是操作系统服务员的角色。



比如在程序中调用 fwrite 函数，图中①，而 fwrite 函数在 glibc 库中调用系统调用 write()（图中②），然后从用户态陷入内核态（图中③），查找系统调用表 syscall table（图中④），在内中中对应的系统调用服务例程为 sys_write,然后在内核执行该例程.



从图可以看出，系统调用要涉及 CPU 状态的转换，首先从用户态陷入到内核态，在内核执行系统调用服务例程，处理结束后，返回到用户态，而一般过程调用，可能在用户态，也可能在内核态，只是一个函数调用另外一个函数，不存在 CPU 状态的转换。

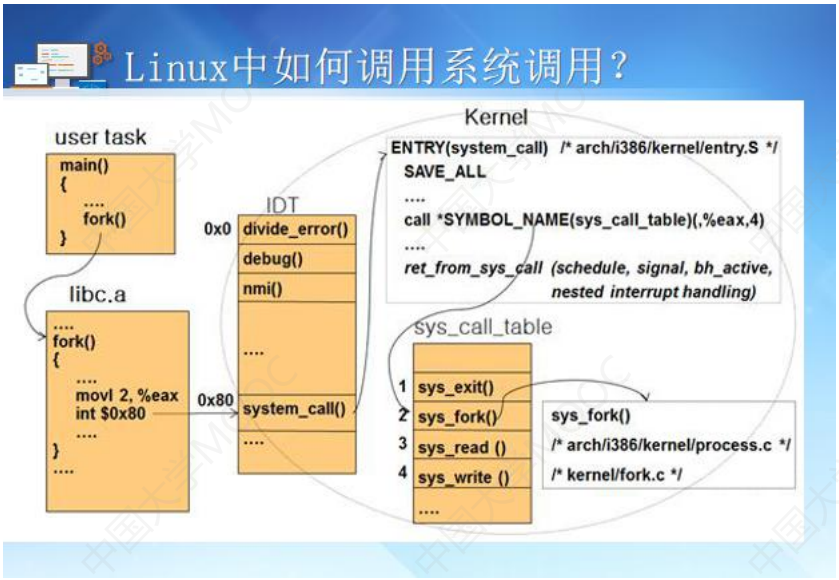
DOS 提供的一组实现特殊功能的子程序，大概有 50 多个，供程序员在编写自己的程序时调用，以减轻编程的工作量。这组系统调用都能干啥,其主要任务有：

(1)负责管理所有的磁盘文件;

(2)负责磁盘空间分配及其他系统资源管理;

(3)负责 DOS 与外层模块的联系等

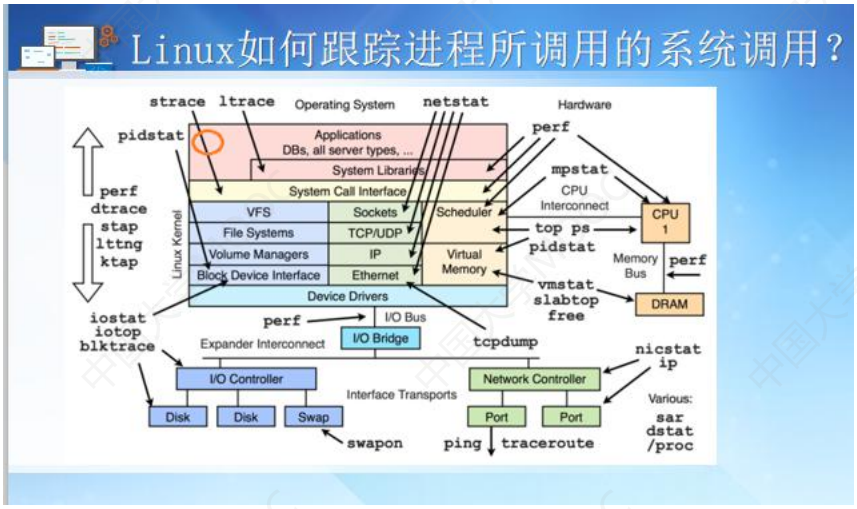
我们看那个 Hello world 的视频，5 行代码就可以在屏幕上输出“hello，world ‘，为什么？实际上，第一行代码是关键，这就是系统调用号，查看系统调用表就知道，它就是在屏幕上显示一行字符串的，第 3~5 行其实就是给入口参数赋值，另外关键的一行就是 int 21H，通过这条指令，就陷入内核了，也就是执行权给内核了。显示 hello world 这件事实际上是内核干的。



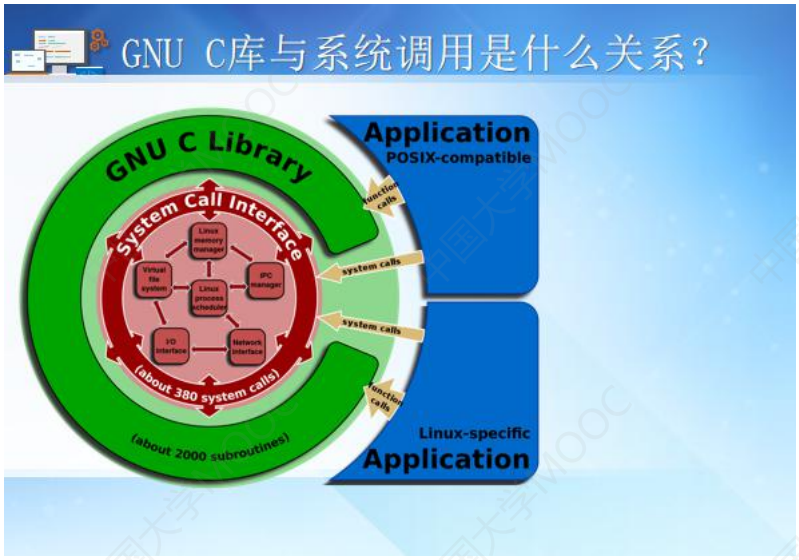
1.从用户程序中调用 fork
2.在 libc 库中把 fork 对应的系统调用号 2 放入寄存器 eax
3.通过 int 0x80 陷入内核
4.在中断描述表 IDT 中查到系统调用的入口 0x80
5.进入 Linux 内核的 entry_32(64).S 文件，从系统调用表 sys_call_table 中找到 sys_fork 的入口地址
6.执行 fork.c 中的 do_fork 代码
7 通过 iret 返回

再换一个系统调用，是不是类似的过程呢，是的，这就是系统调用机制，所谓机制就是像一

个模子一样，大家都套用一个模式。



Linux 下各个子系统都有相应的工具可以对系统进行深入的观察，比如 top 命令，在 Linux 下可以通过 strace 命令查看一个进程到底调用了哪些系统态调用，比如，strace top，你写的程序 myfork，也可以跟踪，strace ./myfork



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.X 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。
这里说一下 POSIX 兼容应用函数调用是什么意思，就是这些函数可以移植到遵循 POSIX 标准的操作系统上，比如 UNIx，但 Linux 专用应用的函数调用就不行，明白了么。

通过前面的介绍，我们知道系统调用就是操作系统提供给用户的服务即接口，把用户从硬件打交道的繁杂事务中解放出来，用户可方便的使用操作系统提供的各种服务。OS 到底是如何处理系统调用的，最重要的是通过陷入指令，达到用户态与系统态之间的切换，同时，为了使得系统调用具有可移植性，业界给出了一个标准，这就是 POSIX 标准，如果想查看一个进程到底调用了哪些系统调用，可以通过是 trace 命令。

二、动手实践

在实验楼发布了通过系统调用创建系统调用的实验，请务必动手实践，并回答下面几个问题：

（1） 在第 12 行执行 fork()时系统进入到什么态？

（2）结合 PPT 中的 fork 的执行流，分析 getpid()的执行流

（3） fork（）的执行对你有什么启发？