9、操作系统的功能

1. 操作系统简介

操作系统是管理计算机硬件和软件资源的计算机程序。操作系统需要处理如管理、调度系统资源（CPU、堆内存等）供需的优先次序、控制输入设备与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

操作系统的种类是多种多样的，不局限于计算机，从手机到超级计算机，操作系统可简单也可复杂，在不同的设备上，操作系统可向用户呈现多种操作。因为我们不可能直接操作计算机硬件，而且设备种类繁多，需要一个统一的界面，因此有了操作系统，操作系统的简易性使得更多人能使用计算机。

如下图1所示，操作系统的基本功能包括三个方面：

1. 操作系统统一管理着计算机资源。这些计算机资源包括处理器资源、存储器资源、IO设备资源和文件资源等。这属于操作系统内核的功能。
2. 操作系统向用户提供了对计算机的操作接口。例如图像窗口形式、命令行形式等。这属于操作系统外壳(Shell)的功能。
3. 操作系统向应用程序提供了对计算机资源调用的接口，程序员无需直接面对硬件接口编程。例如IO设备管理软件，提供读写接口；文件管理软件，提供操作文件的接口。

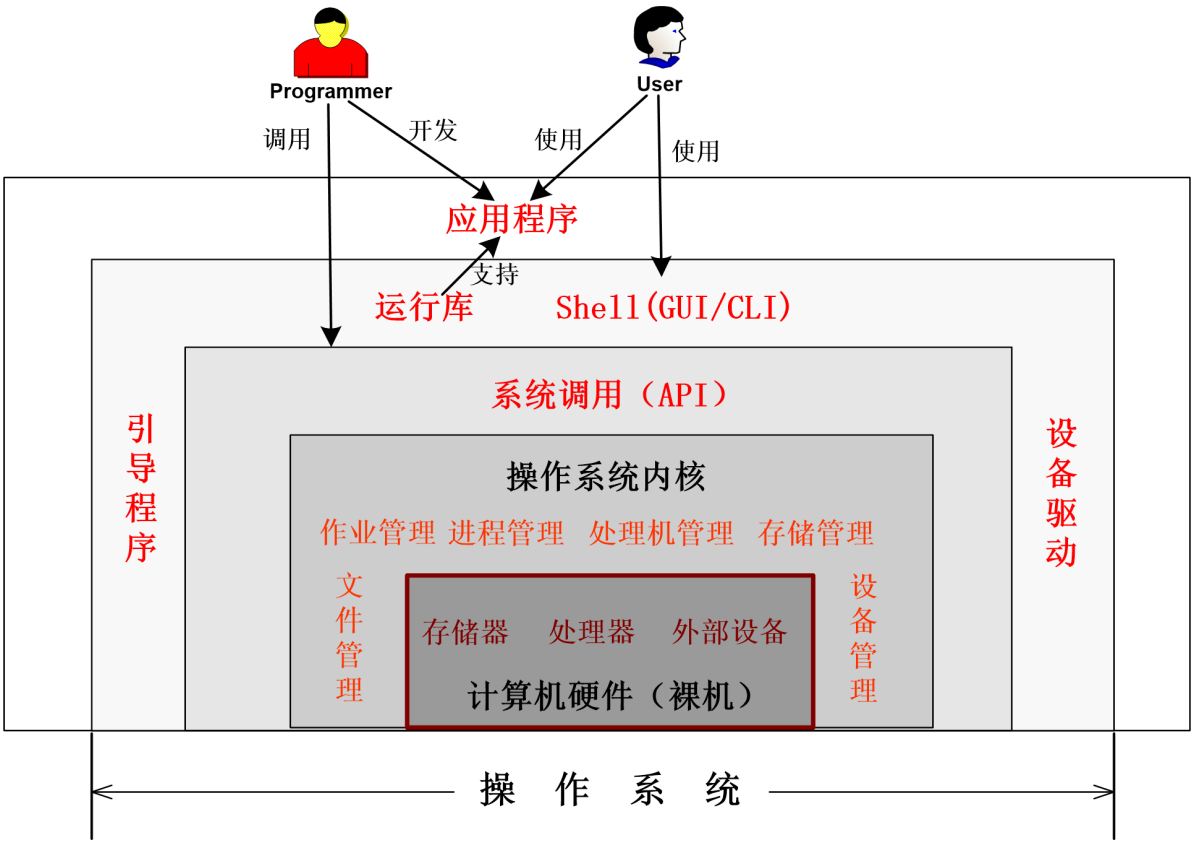

图1、操作系统的基本功能

2. 操作系统内核-资源管理

2.1 处理器管理

处理器管理的基本功能是处理器调度。处理器可能是一个，也可能是多个，不同类型的操作系统将针对不同情况采取不同的调度策略。也叫进程管理。

处理器管理另一个基本的功能是处理中断事件，比如用户的鼠标键盘操作、网络端口事件等。处理器只能发现中断事件并产生中断而不能进行处理。配置了操作系统后，就可对各种事件进行处理。

注：操作系统对中断事件的处理方式深刻影响了交互式应用程序的设计。在课程后面，我们将详细讲解这个问题。

2.2 存储器管理

存储器管理主要是指针对内存储器的管理。主要任务是：将段页线性地址转换到物理地址；分配内存空间，保证各进程占用的存储空间不发生矛盾，并使各进程在自己所属存储区中不互相干扰；进程共享页面。

注：操作系统为作业（应用程序）分配的内存资源是堆（heap），在程序设计时需要关注它，后面我们将进一步讲解这个问题。

2.3 设备管理

设备管理是指负责管理各类外围设备（简称：外设），比如硬盘、U盘、键盘、鼠标、打印机、显示器、网卡等。包括分配、启动和故障处理等。

主要任务是：当用户使用外部设备时，必须提出要求，待操作系统进行统一分配后方可使用。当用户的程序运行到要使用某外设时，由操作系统负责驱动外设。操作系统还具有处理外设中断请求的能力。

2.4 文件管理

操作系统中负责管理和存储文件信息的软件机构称为**文件管理系统**，简称文件系统。文件系统由三部分组成：**文件系统的接口**、对对象操纵和管理的软件集合、对象及属性。

从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

注：我们课程最关注的就是**文件系统的接口**，即应用程序员在程序中操作文件系统的API及其使用方法。将在后面详细介绍。

2.5 进程管理

进程是一个程序对某个数据集的一次执行过程，是分配资源的基本单位。进程也可以说是一系列资源的集和，如cpu时隙，内存，硬盘，句柄等。在进程里面可能跑有一些线程，这些线程依赖这些资源，完成一系列或一个技术性操作。

操作系统进程管理就是指进程创建、进程调度、进程间通信（管道、信号等）等。

2.6 作业管理

进程仅仅是操作系统层面的概念，作业是应用层面的概念。

作业和任务类似，指为达到一定的业务目的，如统计数据并打印，而实施的一系列技术性操作，如链接数据库，执行sql，导出数据，统计整理，格式化，输出到打印机等。

一个作业包括一个或多个进程，几个进程共同完成一个任务。

作业管理包括作业的输入和输出，作业的调度与控制（根据用户的需要控制作业运行的步骤）。

3. 操作系统外壳（Shell）-用户界面

3.1 用户操作计算机

当计算机处于没有操作系统的裸机时代时，用户是通过各种按钮来操作控制计算机，比如输入数据、启动程序、打印结果等。

当计算机安装上操作系统后，用户是如何操作计算机呢？操作系统中专门设计了一个名为Shell的应用程序，充当用户与操作系统交互的界面，即用户通过shell访问并控制计算机。

3.2 从命令行到图形界面

在操作系统的早期，操作系统的用户都是专业用户，比程序员、网管等，并没有图形界面，只有命令行界面，例如linux下的bash、Dos下的command.com、Windows下的cmd.exe（图2）。用户只能通过一个一个的命令来控制计算机。这些命令有成百上千之多，且不说记住这些命令非常困难，每天面对没有任何色彩的“黑屏”本身就是一件枯燥的事情。这个时候的计算机还远远谈不上炫酷和普及，只有专业人员才能使用。

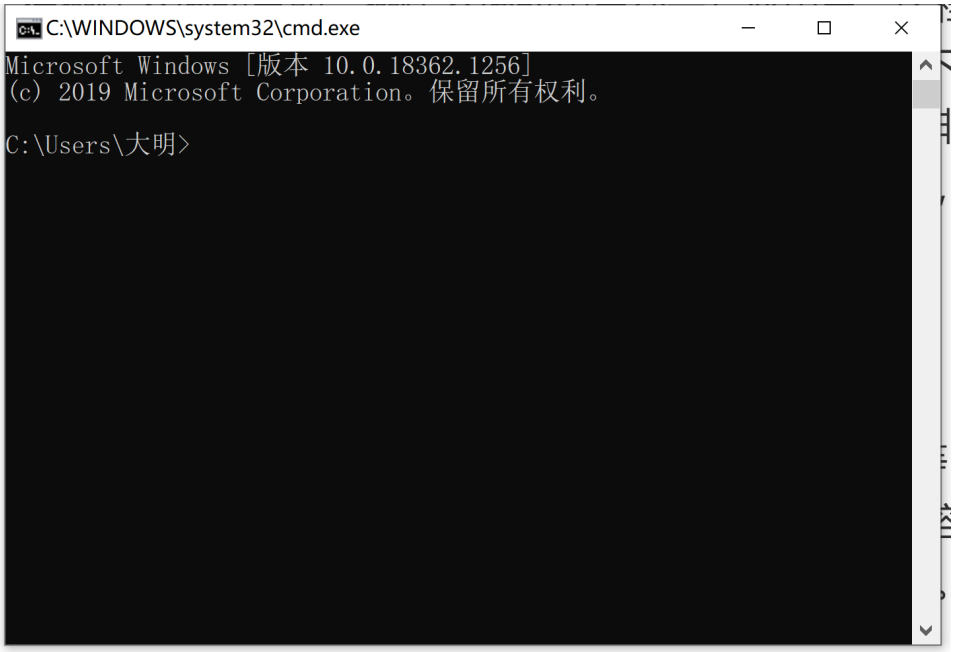

图2、Windows下的命令行cmd

现在操作系统（Windows、Mac OS、Android、iOS 等），我们使用起来很方便，因为它们的Shell都是图形界面的，比如Windows的资源管理器（图3），简单直观，容易上手，对于普通用户（家庭主妇、老年人等）都非常适用。图形界面的应用对计算机的普及起到过巨大的作用。

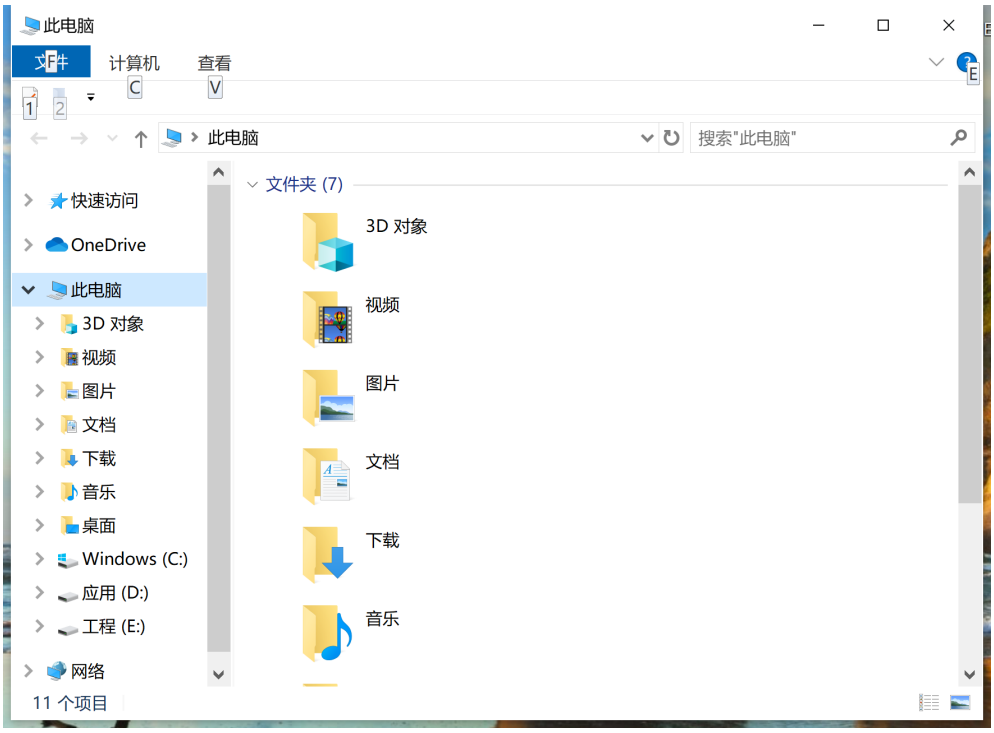

图3、Windows的资源管理器

所以，Shell可以是命令行窗口，也可以是图形界面。

对于图形界面，用户点击某个图标就能启动某个程序。同样地，对于命令行，用户输入某个程序的名字（可以看做一个命令）就能启动某个程序。这两者的基本过程都是类似的，都需要查找程序在硬盘上的安装位置，然后将它们加载到内存运行。

换句话说，图形界面和命令行要达到的目的是一样的，都是让用户控制计算机。

3.3 Shell到底是什么?

Shell是一个随着操作系统启动而运行应用程序，它和 QQ、迅雷、Firefox 等其它软件没有本质的区别。然而 Shell 也有着它的特殊性，就是开机立马启动，并呈现在用户面前。

此外，命令行Shell就是一个命令解释器，能够接收用户输入的命令（附录：Windows下的cmd常用命令），并对命令进行正确解释并处理，执行处理完毕后再将结果反馈给用户，比如输出到显示器、写入到文件等。

图形界面Shell也要负责将用户的鼠标点击命令解释执行。

3.4 外壳与内核

然而，真正能够控制计算机硬件（CPU、内存、显示器等）的只有操作系统内核（Kernel），Shell（图形界面和命令行）只是架设在用户和内核之间的一座桥梁。

为什么要用户不能直接调用内核功能？这是因为安全、复杂等原因，用户不能（也没有必要）直接接触内核。

Shell 程序本身的功能很弱，比如文件操作、输入输出、进程管理等都得依赖内核。Shell的作用就是接收用户的操作（点击图标、输入命令），并进行简单的处理，然后大部分情况下 Shell 都会去调用内核暴露出来的**接口**，这就是在使用内核，只是这个过程被 Shell 隐藏了起来，它自己在背后默默进行，我们看不到而已。

Shell在用户和内核之间增加一层“代理”，既能简化用户的操作，又能保障内核的安全，何乐不为呢?

接口其实就是一个一个的**函数**，使用内核就是调用这些函数。这就是使用内核的全部内容了吗？是的！除了函数，你没有别的途径使用内核。

4. 操作系统接口（API）

用户与计算机交互，是通过操作系统的外壳Shell。

应用程序与计算机交互，是通过操作系统的接口—API。

4.1 什么是接口?

首先，我们从日常生活中的一个小例子来了解**接口**的概念。

图4、电源接口

图4的插座就是一个接口，插座的里面连接着线路，插座的外面连接着我们所使用的电器的插头。

有了插座以后，我们只需要把电器的插头插进插座，而不需要关注插座后面的复杂连接，便可以使用我们的电器了。

因此，我们很直观地认识到接口的含义：**连接两个东西**(插座建立了里面的线路和外面的插头的连接)，**屏蔽细节**(插座里面连接的复杂线路用户不需要知道)，**方便用户使用**(插上插座就能用)。

4.2 操作系统接口和系统调用

类似上面的例子，我们看看操作系统的接口是什么。

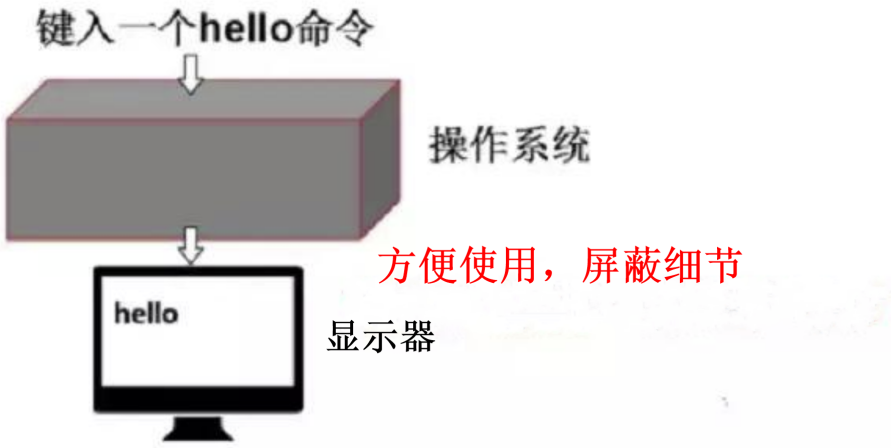

图5、操作系统接口与使用示例

同样地，操作系统接口也具有连接两个东西、屏蔽细节、方便用户使用的特点。它连接上层应用软件和底层硬件，屏蔽细节，用户直接通过程序(应用软件)使用计算机，方便用户使用。

如上图5所示，我们只需要通过在键盘上敲一个hello的命令(应用程序代码)，该命令通过操作系统的内部处理，处理后在显示屏上显示hello字样。操作系统就相当于一个黑盒子，我们无需关注内部实现，对外面的用户屏蔽其内部细节。

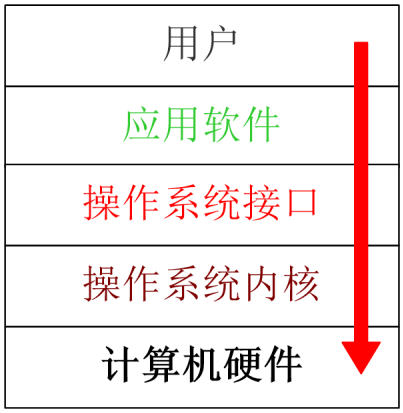

图6、计算机调用层次

如图6所示，用户使用应用程序，应用程序调用操作系统接口，操作系统接口调用操作系统内核，操作系统内核控制计算机硬件。

其实，**操作系统向应用程序提供的接口就是一些C语言函数**，我们程序员在编写的程序中**去调用这些操作系统函数**，完成**对底层硬件的控制。**

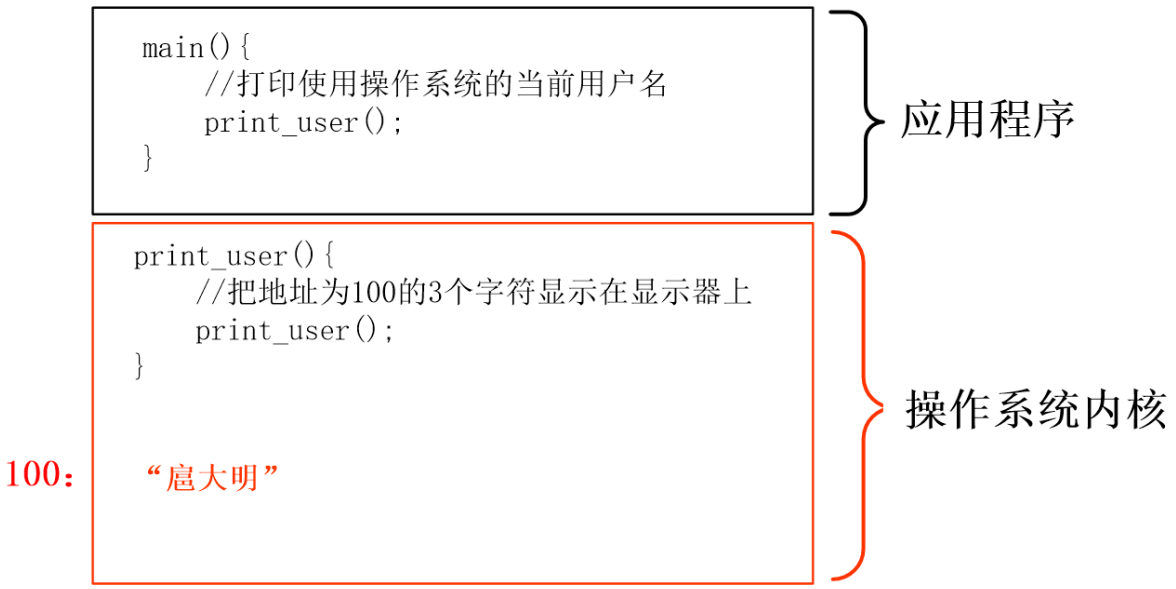

图7、系统调用

图7上端黑色表示是应用程序区，下端红色表示操作系统内核。操作系统内核有一个打印操作系统当前用户名的函数 print_user()，用户程序通过调用该print_user() 函数，实现该用户程序在屏幕上打印出操作系统当前使用者名字的功能。其中，print_user()就是操作系统的接口，上层的用户程序通过调用该接口，该接口完成内部完成打印功能。

这就是操作系统的接口了，它表现为函数调用，又由于它由操作系统提供，所以称为系统调用。

4.3 系统调用层的必要性

继续讨论图7，存在一个问题：

> 那我们为什么要通过系统调用而不能直接去访问操作系统内核内存地址为100的数据?

因为操作系统内核存放着有关于操作系统、计算机软硬件等各种重要的数据，比如操作系统root用户的密码，如果我们直接访问系统内核，就可以随便看到root用户的密码，还可以去修改它。又比如通过内核里面的显存信息能看到别人word里内容，这样就不安全了！所以操作系统封装一些功能接口，然后让用户去调用这些接口而禁止用户直接访问内核中的数据。如果能直接访问就能去修改一些数据，就能随便查看一些敏感的数据，这样对用户非常不安全。

所以操作系统把内存分为了操作系统内核段和用户程序用户段，把在内核段执行的代码和数据称为处于内核态，把在用户段执行的代码和数据称为处于用户态，将内核程序和用户程序隔离！使得内核态可以访问任何数据，用户态不能访问内核段数据而只能访问用户段数据。

4.4 内核态、用户态

继续研究图7。

图7其实是内存模型。上端黑色区域为用户段，执行普通用户程序；下端红色区域为内核段，执行操作系统内核代码。

如果此时CPU在处理上端黑色区域的 main() 函数，那此时处于用户态。如果此时CPU在执行下端红色区域的 print_user() 函数，那此时处于内核态。

1）处于内核态可以访问用户段和内核段的数据。

2）处于用户态只能访问用户段的数据而不能访问内核段的数据。

那么如何区分内核态和用户态呢?

计算机采取了一种处理器的"硬件设计"去区分。完成这功能的要用到两个寄存器，分别是CPL寄存器和DPL寄存器。

CPL寄存器表示当前程序执行在什么态，0表示内核态，3表示用户态；

DPL寄存器表示即将访问的数据在什么段，同样0表示内核段，3表示用户段。

每次访问数据的时候检查两个寄存器的大小关系，若DPL≥CPL，则可以访问；反之，则不能访问。

一、例如当前程序运行在用户态，那么此时的CPL为3，若即将访问的数据在用户段（DPL为3），此时满足DPL≥CPL，可以访问（即用户态访问用户段数据），若即将访问的数据在内核段（DPL为0），则此时DPL<CPL，因此不能访问。

二、同理，假如当前程序运行在内核态，那么此时的CPL为0，因此无论即将访问的数据是处于内核段（DPL为0）还是用户段（DPL为3），都满足DPL≥CPL，因此处于内核态的程序可以访问任何数据。

CPL为0表示在内核段，CPL为3表示在用户段，那么CPL为1或2表示什么呢？参见下图8：

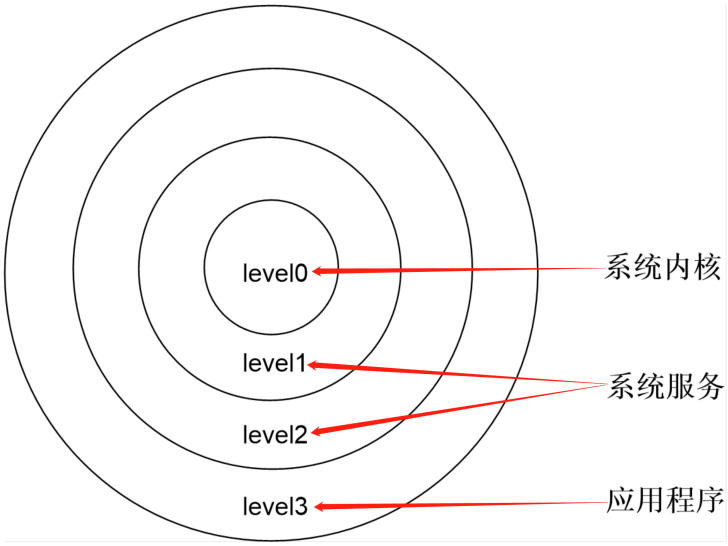

图8、内核-服务-应用

附录：Windows下cmd的常用命令

开始 - >运行 - >键入cmd

cd \　　//跳转到硬盘的根目录

cd C:\WINDOWS　//跳转到当前硬盘的其他文件

d:　　//跳转到其他硬盘

cd /d e:\software　//跳转到其他硬盘的其他文件夹

cd..　//跳转到上一层目录

cd /?　//获取使用帮助

ping -help

ping ip(或域名)　//向对方主机发送默认大小为32字节的数据

ipconfig #查看自己的ip

netstat -ano //查看网络连接、状态以及对应的进程id

netstat -a　//查看开启了哪些端口,常用netstat -an

netstat -n　//查看端口的网络连接情况，常用netstat -an

netstat -v　//查看正在进行的工作

find 文件名 //查找某文件

md 目录名 //创建目录

replace 源文件 要替换文件的目录 //替换文件

ren 原文件名 新文件名 //重命名文件名

.....