Linux 内核的属性

在讨论大型而复杂的系统的体系结构时，可以从很多角度来审视系统。体系结构分析的一个目标是提供一种方法更好地理解源代码。

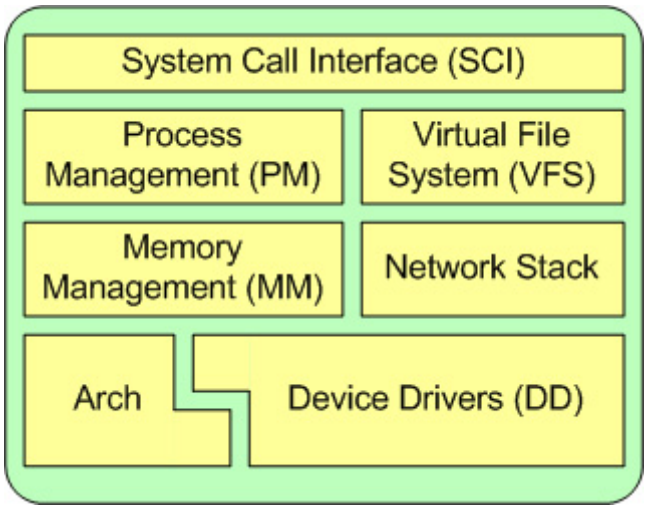
Linux 内核实现了很多重要的体系结构属性。在或高或低的层次上，内核被划分为多个子系统。Linux 也可以看作是一个整体，因为它会将所有这些基本服务都集成到内核中。这与微内核的体系结构不同，后者会提供一些基本的服务，例如通信、I/O、内存和进程管理，更具体的服务都是插入到微内核层中的。每种内核都有自己的优点，不过这里并不对此进行讨论。

随着时间的流逝，Linux 内核在内存和 CPU 使用方面具有较高的效率，并且非常稳定。但是对于 Linux 来说，最为有趣的是在这种大小和复杂性的前提下，依然具有良好的可移植性。Linux 编译后可在大量处理器和具有不同体系结构约束和需求的平台上运行。一个例子是 Linux 可以在一个具有内存管理单元(MMU)的处理器上运行，也可以在那些不提供 MMU 的处理器上运行。Linux 内核的 uClinux 移植提供了对非 MMU 的支持。更详细信息请参看 参考资料 一节的内容。

1、 Linux 内核的主要子系统

现在使用图 3 中的分类说明 Linux 内核的主要组件。

下图是 Linux 内核的一个体系结构透视图。



A. 系统调用接口

SCI 层提供了某些机制执行从用户空间到内核的函数调用。正如前面讨论的一样，这个接口依赖于体系结构，甚至在相同的处理器家族内也是如此。SCI 实际上是一个非常有用的函数调用多路复用和多路分解服务。在 ./linux/kernel 中您可以找到 SCI 的实现，并在 ./linux/arch 中找到依赖于体系结构的部分。

B. 进程管理

进程管理的重点是进程的执行。在内核中，这些进程称为线程，代表了单独的处理器虚拟化（线程代码、数据、堆栈和 CPU 寄存器）。在用户空间，通常使用进程 这个术语，不过 Linux 实现并没有区分这两个概念（进程和线程）。内核通过 SCI 提供了一个应用程序编程接口(API)来创建一个新进程(fork、exec 或 Portable Operating System Interface [POSIX] 函数)，停止进程(kill、exit)，并在它们之间进行通信和同步(signal 或者 POSIX 机制）。

进程管理还包括处理活动进程之间共享 CPU 的需求。内核实现了一种新型的调度算法，不管有多少个线程在竞争 CPU，这种算法都可以在固定时间内进行操作。这种算法就称为

O(1) 调度程序，这个名字就表示它调度多个线程所使用的时间和调度一个线程所使用的时间是相同的。 O(1) 调度程序也可以支持多处理器（称为对称多处理器或 SMP）。您可以在 ./linux/kernel 中找到进程管理的源代码，在 ./linux/arch 中可以找到依赖于体系结构的源代码。在 参考资料 一节中可以了解有关这个算法的更多内容。

C. 内存管理

内核所管理的另外一个重要资源是内存。为了提高效率，如果由硬件管理虚拟内存，内存是按照所谓的内存页 方式进行管理的（对于大部分体系结构来说都是 4KB）。Linux 包括了管理可用内存的方式，以及物理和虚拟映射所使用的硬件机制。

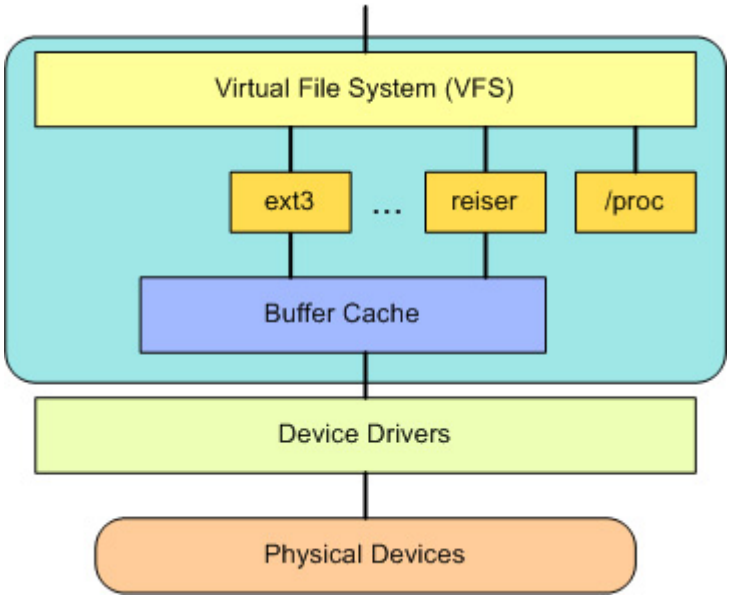
不过内存管理要管理的可不止 4KB 缓冲区。Linux 提供了对 4KB 缓冲区的抽象，例如 slab 分配器。这种内存管理模式使用 4KB 缓冲区为基数，然后从中分配结构，并跟踪内存页使用情况，比如哪些内存页是满的，哪些页面没有完全使用，哪些页面为空。这样就允许该模式根据系统需要来动态调整内存使用。

为了支持多个用户使用内存，有时会出现可用内存被消耗光的情况。由于这个原因，页面可以移出内存并放入磁盘中。这个过程称为交换，因为页面会被从内存交换到硬盘上。内存管理的源代码可以在 ./linux/mm 中找到。

D. 虚拟文件系统

虚拟文件系统（VFS）是 Linux 内核中非常有用的一个方面，因为它为文件系统提供了一个通用的接口抽象。VFS 在 SCI 和内核所支持的文件系统之间提供了一个交换层

下图在用户和文件系统之间提供了一个交换层。



在 VFS 上面，是对诸如 open、close、read 和 write 之类的函数的一个通用 API 抽象。在 VFS 下面是文件系统抽象，它定义了上层函数的实现方式。它们是给定文件系统（超过 50 个）的插件。文件系统的源代码可以在 ./linux/fs 中找到。

文件系统层之下是缓冲区缓存，它为文件系统层提供了一个通用函数集（与具体文件系统无关）。这个缓存层通过将数据保留一段时间（或者随即预先读取数据以便在需要是就可用）优化了对物理设备的访问。缓冲区缓存之下是设备驱动程序，它实现了特定物理设备的接口。

E. 网络堆栈

网络堆栈在设计上遵循模拟协议本身的分层体系结构。回想一下，Internet Protocol (IP) 是传输协议（通常称为传输控制协议或 TCP）下面的核心网络层协议。TCP 上面是 socket 层，它是通过 SCI 进行调用的。

socket 层是网络子系统的标准 API，它为各种网络协议提供了一个用户接口。从原始帧访问到 IP 协议数据单元（PDU），再到 TCP 和 User Datagram Protocol (UDP)，socket 层提供了一种标准化的方法来管理连接，并在各个终点之间移动数据。内核中网络源代码可以在 ./linux/net 中找到。

F. 设备驱动程序

Linux 内核中有大量代码都在设备驱动程序中，它们能够运转特定的硬件设备。Linux 源码树提供了一个驱动程序子目录，这个目录又进一步划分为各种支持设备，例如 Bluetooth、I2C、serial 等。设备驱动程序的代码可以在 ./linux/drivers 中找到。

G. 依赖体系结构的代码

尽管 Linux 很大程度上独立于所运行的体系结构，但是有些元素则必须考虑体系结构才能正常操作并实现更高效率。./linux/arch 子目录定义了内核源代码中依赖于体系结构的部分，其中包含了各种特定于体系结构的子目录（共同组成了 BSP）。对于一个典型的桌面系统来说，使用的是 i386 目录。每个体系结构子目录都包含了很多其他子目录，每个子目录都关注内核中的一个特定方面，例如引导、内核、内存管理等。这些依赖体系结构的代码可以在 ./linux/arch 中找到。

2、 Linux 内核的一些有用特性

如果 Linux 内核的可移植性和效率还不够好，Linux 还提供了其他一些特性，它们无法划分到上面的分类中。

作为一个生产操作系统和开源软件，Linux 是测试新协议及其增强的良好平台。Linux 支持大量网络协议，包括典型的 TCP/IP，以及高速网络的扩展（大于 1 Gigabit Ethernet [GbE] 和 10 GbE）。Linux 也可以支持诸如流控制传输协议（SCTP）之类的协议，它提供了很多比 TCP 更高级的特性（是传输层协议的接替者）。

Linux 还是一个动态内核，支持动态添加或删除软件组件。被称为动态可加载内核模块，它们可以在引导时根据需要（当前特定设备需要这个模块）或在任何时候由用户插入。

Linux 最新的一个增强是可以用作其他操作系统的操作系统（称为系统管理程序）。最近，对内核进行了修改，称为基于内核的虚拟机（KVM）。这个修改为用户空间启用了一个新的接口，它可以允许其他操作系统在启用了 KVM 的内核之上运行。除了运行 Linux 的其他实例之外， Microsoft® Windows® 也可以进行虚拟化。惟一的限制是底层处理器必须支持新的虚拟化指令。