Unix 考试复习大纲(2018.11.8)

**1.UNIX操作系统最根本的功能特征是什么？包括哪些最基本的概念？**

分时多用户、开放性。

分时多用户：多个用户多个进程同时在一个系统中运行，系统资源高度共享、有效协调

并发开放性：标准化，结构上的一致性

可移植性：应用软件的编码及系统应用接口

可互操作性：可保持用户原来的使用习惯、异种机之间的互操作

(1).交互式分时多用户：人机间实时交互数据, 多个用户可同时使用一台机器, 每个用户可同时执行多个任务

(2).软件复用：每个程序模块完成单一的功能, 程序模块可按需任意组合, 较高的系统和应用开发效率

(3).可移植性强：数千行汇编码, 数十万行C语言代码

(4).配置灵活, 适应性强：小内核, 参数灵活可调,核外应用系统, 任意裁减, 限制规则很少

(5). 界面方便高效：内部: 系统调用丰富高效,外部: shell命令灵活方便可编程,应用: GUI 清晰直观功能强大

(6 )安全机制完善：口令、权限、加密等措施完善, 抗病毒结构,误操作的局限和自动恢复功能

(7).多国语言支持, 支持全世界现有的几十种主要语言

(8).网络和资源共享：内部: 多进程结构易于资源共享, 外部: 支持多种网络协议

**2、操作系统核心是什么？核心通过什么方式和什么原则向上层应用程序提供了哪些服务？**

系统调用的集合及实现系统调用的内部算法就形成操作系统核心。

内核必须要提供的功能：中断处理、短程调度、原语管理

以函数形式提供给核外的命令和上层应用系统使用的一组程序,涵盖操作系统的所有功能。是应用程序请求操作系统服务的唯一通道。

**3、数据缓冲区高速缓冲是建立的基础和原则是什么？要解决的根本问题是什么？他有什么优缺点？**

(1)基础与原则是:

对于文件系统的一切存取操作，内核都能通过每次直接从磁盘上读或往磁盘上写来实现。但是，慢的磁盘传输速率会使系统响应时间加长、吞吐率降低。内核通过保持一个称为数据缓冲区高速缓冲的内部数据缓冲池来试图减少对磁盘的存取频率。

(2)解决的根本问题是:

磁盘机械运行速度大大低于处理机的运行速度;多进程并发运行,少量的磁盘(通道),I/O成为瓶颈;数据访问的随机性,磁盘忙闲不均。

(3)数据高速缓冲区优缺点：

优点:

提供了对磁盘块的统一的存取方法; 消除了用户对用户缓冲区中数据的特殊对齐需要;减少了磁盘访问的次数,提高了系统的整体I/O效率;有助于保持文件系统的完整性。

缺点：

数据未及时写盘而带来的风险;额外的数据拷贝过程,大量数据传输时影响性能。

1. **中断与例外分别是哪里产生的？他们的根本区别是什么？他们的处理方式和流程有何区别？中断级别是根据什么来划分的？为什么要在系统中设置不同的处理机执行级别？**

(1)中断：来自进程之外的事件(外设、时钟等)引起的。

例外: 来自进程内部的非期望事件(地址越界,除数为0等)。

1. 根本区别：中断要保存上下文,发生在两条指令执行之间；例外不保存上下文，发生在一条指令执行过程中。

(3).处理方式和流程的区别:

中断：当接收到中断的时候，内核保存它的当前上下文，判定中断原因，为中断服务。当中断服务完毕后从下一条指令继续执行(中断服务是由核心中特殊的函数,而不是特殊的进程来执行的)。

例外: 来自进程内部的非期望事件(地址越界,执行特权指令除数为0等),发生在一条指令执行过程中,系统在例外事件处理完后重新执行该指令。

|  |  |
| --- | --- |
| 中断事件 | 中断级别 |
| 硬盘故障 | 高 |
| 时钟 |  |
| 硬盘 |  |
| 网路 |  |
| 终端 |  |
| 软件中断 | 低 |

(4).中断级别的划分:

(5).不同级别？

用一组特权指令给处理机设置一个执行级,以屏蔽同级和低级的中断,最大限度地减少其它事件的干扰,使当前任务顺利执行并尽快完成;但开放更高级的中断,以响应更紧迫的请求。

**5、操作系统中包括了哪些构建原语？他们是如何构建更大的功能模块的？**

一般地，把系统态下执行的某些具有特定功能的程序段成为原语。一类是[机器指令](https://www.baidu.com/s?wd=%E6%9C%BA%E5%99%A8%E6%8C%87%E4%BB%A4&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "/home/wtl/文档\\x/_blank)级的，其特点是执行期间不允许中断，正如在物理学中的原子一样，在[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "/home/wtl/文档\\x/_blank)中是一个不可分割的基本单位;  
另一类是功能型的，其特点是作为原语的程序段不允许并发执行。  
 原语均在系统态下执行，且都是为了完成某个系统管理所需要的功能被高层软件所调用，具有不可分割性；即原语的执行必须是连续的，在执行过程中不允许被中断。

两种构建原语：1 输入/输出重定向。2 管道。

进程相关：创建进程原语Create()、撤销原语destroy( )、挂起原语suspend()、激活原语active( )、唤醒原语 wakeup()

进程管理、主存储器管理、文件管理、I/O系统管理、辅助存储器管理、网络管理、系统保护

操作系统的基本服务：程序执行、I/O操作、文件系统处理、通信、错误检测

**6、UNIX的文件系统包括了哪些大的功能模块？什么是本地文件系统？什么是虚拟文件系统？设置虚拟文件系统的优缺点是什么？**

**超级块：**文件系统中第一块被称为超级块。这个块存放文件系统本身的结构信息，比如每个区域的大小，未被使用的磁盘块的信息。

**i节点：**超级块的下一个部分就是i节点表，文件系统中的每个文件在该表中都对应一个i节点。i节点是固定长度的记录项，它包含有关文件的大部分信息。Linux文件系统使用 索引节点i 来记录文件信息，索引节点是一个结构，用固定长度，它包含了一个文件的长度、创建及修改时间、权限、所属关系、磁盘中的位置等信息。

**数据区：**文件的内容保存在这个区域上，磁盘上所有块的大小都一样，如果文件包含了超过一个块的内容，则文件内容会存放在多个磁盘块中，并把磁盘块的分配情况记录在文件的i节点中的磁盘序列表中

参考：<http://www.cnblogs.com/qianye/archive/2012/11/24/2786345.html>

本地文件系统是UNIX系统中的基本文件系统,它通常固定存放在本地机器的存贮设备上,任何一种结构形式的文件系统都必然会直接或间接地与某个本地文件系统相联系。本地文件系统由一个根文件系统 和 若干子文件系统所组成。

虚拟文件系统是整个操作系统的用户界面,它给用户提供一个统一的文件系统使用接口,避免用户涉及各个子文件系统的特征部分。用户感觉使用的是一个整体的,比本地机器上实际硬盘空间大得多的文件系统。虚构文件系统接受来自用户的操作请求,根据该操作所访问的文件是存放在本地机器上,还是存放在远地机器上而分别把操作交给本地文件系统或网络文件系统;本地文件系统或网络文件系统(实际上再传给远地机器上的本地文件系统)进行相应的操作后,将结果返回到虚拟文件系统中再传回给用户。

优点：使得整个文件系统结构统一,模块性强,增加功能非常方便;对用户来说,整个文件系统的透明性好,使用简便,避免了用户在对不同类型的文件或不同地点的文件系统进行操作时,分别来设置参数和安排操作过程。

缺点:PVFS中应用系统socket相互通信，应用TCP/IP通信协议，每次通信需要内核嵌入，进行内存拷贝，CPU的负载比较大，影响系统的运行效率；PVFS并行虚拟文件系统本身具备良好的可扩展性，但是其动态配置的能力不强，如果要扩展一个I/O节点，就需要停止服务，并且不能做到空间的合理利用等。

参考：<https://blog.csdn.net/foreverdengwei/article/details/8476008>

1. **划分数据块和数据片的目的是什么？**
2. **资源保护系统以什么方式保护了哪些类型的资源？**

共享资源: 中断屏蔽、原子操作、自旋锁、信号量、环形缓冲区、互斥体

内存防护：checksec 用来检查可执行文件属性、CANNARY(栈溢出保护)是一种缓冲区溢出攻击缓解手段、内存地址随机化机制（address space layout randomization)

参考：<https://blog.csdn.net/hudaweikevin/article/details/1687982>

<https://www.cnblogs.com/Spider-spiders/p/8798628.html>

1. **逻辑地址和物理地址的区别是什么？他们是怎么转换的？**

**物理地址：**CPU地址总线传来的地址，由硬件电路控制（现在这些硬件是可编程的了）其具体含义。物理地址中很大一部分是留给内存条中的内存的，但也常被映射到其他存储器上（如显存、BIOS等）。在没有使用虚拟存储器的机器上，虚拟地址被直接送到内存总线上，使具有相同地址的物理存储器被读写；而在使用了虚拟存储器的情况下，虚拟地址不是被直接送到内存地址总线上，而是送到存储器管理单元MMU，把虚拟地址映射为物理地址。

**线性地址**（Linear Address）也叫虚拟地址(virtual address)是逻辑地址到物理地址变换之间的中间层。在分段部件中逻辑地址是段中的偏移地址，然后加上基地址就是线性地址。是一个32位无符号整数，可以用来表示高达4GB的地址，也就是，高达4294967296个内存单元。线性地址通常用十六进制数字表示，值得范围从0x00000000到0xfffffff）程序代码会产生逻辑地址，通过逻辑地址变换就可以生成一个线性地址。如果启用了分页机制，那么线性地址可以再经过变换以产生一个物理地址。如果没有启用分页机制，那么线性地址直接就是物理地址。

**逻辑地址**：在有地址变换功能的计算机中,访内指令给出的地址 (操作数) 叫逻辑地址,也叫相对地址，也就是是机器语言指令中，用来指定一个操作数或是一条指令的地址。要经过寻址方式的计算或变换才得到内存储器中的实际有效地址即物理地址。一个逻辑地址由两部份组成，段标识符: 段内偏移量。段标识符是由一个16位长的字段组成，称为段选择符。其中前13位是个索引号，后面3位包含一些硬件细节 。

CPU将一个逻辑地址转换为物理地址，需要进行两步：首先将给定一个逻辑地址（其实是段内偏移量，这个一定要理解！！！），CPU要利用其段式内存管理单元，先将为个逻辑地址转换成一个线程地址，再利用其页式内存管理单元，转换为最终物理地址。

参考：<https://blog.csdn.net/prike/article/details/52722934>

1. **文件的i节点有什么样的特点和功能？**

当一个文件被创建时，系统会分配一个inode给它，这样就将文件名和inode关联起来了，我们可以将inode看做文件系统的内部名称。当用户或程序引用该文件时，系统就会通过该文件的文件名来查找相应的inode，然后通过inode来获取所需要的文件信息。也就是说在UNIX系统中文件名仅仅只是与inode相连，而不是与文件相连，这是和Windows系统中不同的一点。

i节点是对文件进行控制和管理的一种数据结构。每一个文件都有自己的i节点，每个i节点都有一个唯一的i节点号。i节点结构如下(参考/usr/include/sys/ino.h)：

struct dinode

{

ushort di\_mode;  /\*文件类型+用户权限\*/

short di\_nlink;  /\*文件链接数\*/

ushort di\_uid;  /\*属主用户id\*/

ushort di\_gid;  /\*属主用户组id\*/

off\_t di\_size;  /\*文件大小\*/

char di\_addr[40]; /\*文件数据区起点地址\*/

time\_t di\_atime; /\*最后访问时间\*/

time\_t di\_mtime; /\*最后修改时间\*/

time\_t di\_ctime; /\*创建时间\*/

};

从上面这个结构可以看出以下一些信息：

1、i节点保存了文件的属性和类型、存放文件内容的物理块地址、最近一次的存取时间、最近一次的修改时间、创建此文件的时间。

2、i节点中没有记录文件名字，那文件名是怎么关联到i节点

3、di\_mode保存文件类型+用户权限。

参考：<https://blog.csdn.net/u011403897/article/details/9526237>

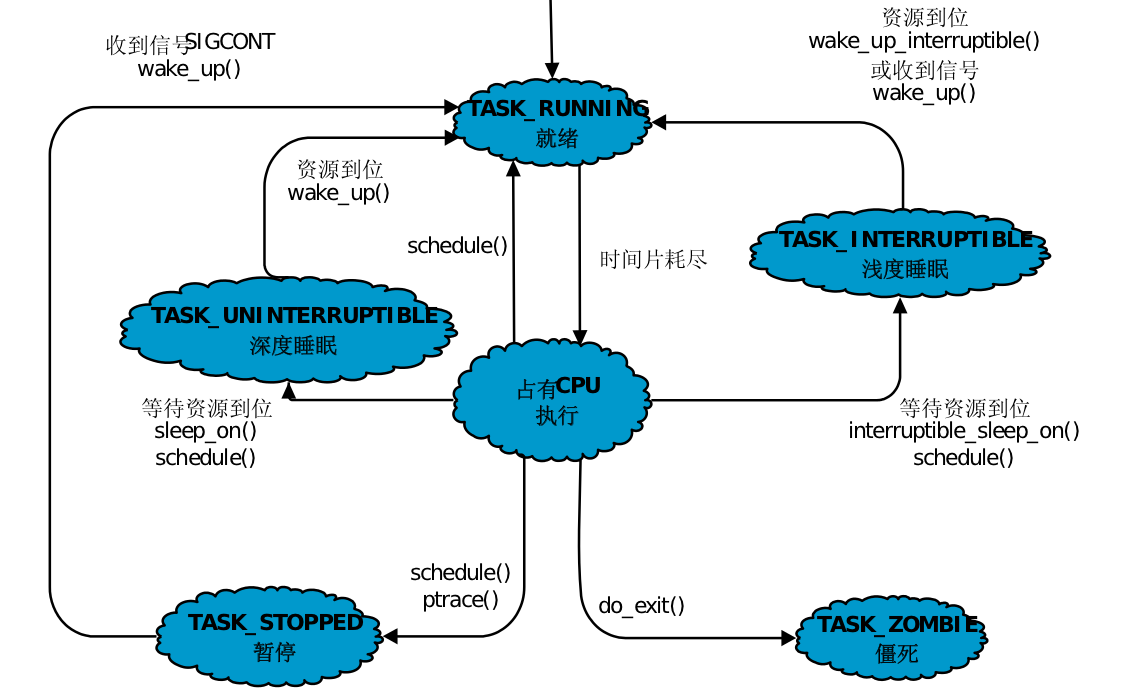
1. **进程的三个最基本的进程状态？进程的生命周期中可能要经过哪些过程？**

运行态：进程占用CPU，并在CPU上运行；

就绪态：进程已经具备运行条件，但是CPU还没有分配过来；

阻塞态：进程因等待某件事发生而暂时不能运行；

LINUX进程间状态转换和内核调用图解：



运行状态（TASK\_RUNNING）：意味着进程处于可运行状态。并不意味着已经实际分配了CPU。进程可能会一直等到调度器选中它。该状态确保进程可以立即运行，而无需等待外部事件。

可中断睡眠状态（浅度睡眠）（TASK\_INTERRUPTIBLE）：针对等待某事件或其他资源的睡眠进程设置的。在内核发送信号给该进程表面事件已经发生时，进程状态变为TASK\_RUNNING，它只要调度器选中该进程即可恢复执行。

不可中断睡眠状态（深度睡眠状态）（TASK\_UNINTERRUPTIBLE）：其和浅度睡眠基本类似，因内核指示而停用的睡眠进程。不能由外部信号唤醒，只能由内核亲自唤醒。

暂停状态（TASK\_STOPPED）：进程暂停执行接受某种处理。如正在接受调试的进程处于这种状态僵死状态（TASK\_ZOMBIE）：进程已经结束但未释放PCB

参考：<https://blog.csdn.net/maliao1123/article/details/54973884>

<https://blog.csdn.net/qq_29168493/article/details/79386139>

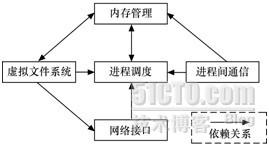
1. **标准输入输出重定向的基本原理是什么？如何来实现标准输入输出重定向？**

理解I/O重定向的原理需要从Linux内核为进程所维护的关键数据结构入手。对Linux进程来讲，每个打开的文件都是通过文件描述符(File Descriptor)来标识的，内核为每个进程维护了一个文件描述符表，这个表以FD为索引，再进一步指向文件的详细信息。在进程创建时，内核为进程默认创建了0、1、2三个特殊的FD，这就是STDIN、STDOUT和STDERR。所谓的I/O重定向也就是让已创建的FD指向其他文件。在I/O重定向的过程中，不变的是FD 0/1/2代表STDIN/STDOUT/STDERR，变化的是文件描述符表中FD 0/1/2对应的具体文件，应用程序只关心前者。本质上这和接口的原理是相通的，通过一个间接层把功能的使用者和提供者解耦。

1. 首先fork一个子进程，后续步骤都在子进程中完成，父进程通过waitpid()系统调用等待子进程结束；
2. 打开open()系统调用打开in.txt和out.txt，得到它们的描述符;
3. 通过dup2()系统调用把STDIN重定向到fd\_in，把STDOUT重定向到fd\_out（注意，重定向的影响范围是整个子进程）；

dup2(fd\_in, 0); dup2(fd\_out, 1);

参考：<http://www.cnblogs.com/weidagang2046/p/io-redirection.html>

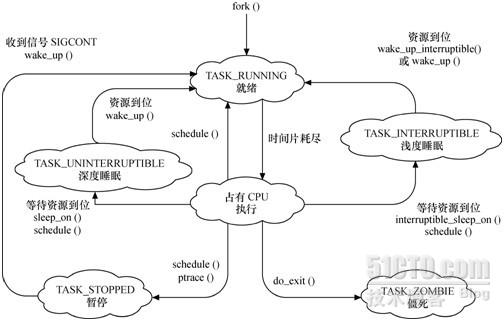
体系的上部分是用户（或应用程序）空间，这是用户应用程序执行的地方。用户空间之下是内核空间，Linux 内核正是位于这里。Linux 内核可以进一步划分成 3 层：最上面是系统调用接口，用户程序通过软件中断后，调用系统内核提供的功能，这个在用户空间和内核提供的服务之间的接口称为系统调用，它实现了一些基本的功能，例如 read 和 write；系统调用接口之下是内核代码，可以更精确地定义为独立于体系结构的内核代码，这些代码是 Linux 所支持的所有处理器体系结构所通用的；内核代码之下是依赖于体系结构的代码，构成了通常称为 BSP（Board Support Package）的部分，这些代码用作给定体系结构的处理器和特定于平台的代码。

然后介绍Linux内核的五个子系统：

Linux内核主要由进程调度（SCHED）、内存管理（MM）、虚拟文件系统（VFS）、网络接口（NET）和进程间通信（IPC）5个子系统组成，如图：

1．进程调度

进程调度控制系统中的多个进程对CPU的访问，使得多个进程能在CPU中“微观串行，宏观并行”地执行。进程调度处于系统的中心位置，内核中其他的子系统都依赖它，因为每个子系统都需要挂起或恢复进程。

 如上图2所示，Linux的进程在几个状态间进行切换。在设备驱动编程中，当请求的资源不能得到满足时，驱动一般会调度其他进程执行，并使本进程进入睡眠状态，直到它请求的资源被释放，才会被唤醒而进入就绪态。睡眠分成可被打断的睡眠和不可被打断的睡眠，两者的区别在于可被打断的睡眠在收到信号的时候会醒。

在设备驱动编程中，当请求的资源不能得到满足时，驱动一般会调度其他进程执行，其对应进程进入睡眠状态，直到它请求的资源被释放，才会被唤醒而进入就绪态。设备驱动中，如果需要几个并发执行的任务，可以启动内核线程，启动内核线程的函数为：

pid\_t kernel\_thread(int (\*fn)(void \*), void \*arg, unsigned long flags);

当用户使用系统提供的库函数进行进程编程，用户可以动态地创建进程，进程之间还有等待，互斥等操作，这些操作都是由linux内核来实现的。linux内核通过进程管理子系统实现了进程有关的操作，在linux系统上，所有的计算工作都是通过进程表现的，进程可以是短期的（执行一个命令），也可以是长期的（一种网络服务）。linux系统是一种动态系统，通过进程管理能够适应不断变化的计算需求。

在用户空间，进程是由进程标示符（PID）表示的。从用户角度看，一个PID是一个数字值，可以唯一标识一个进程，一个PID值在进程的整个生命周期中不会更改，但是PID可以在进程销毁后被重新使用。创建进程可以使用几种方式，可以创建一个新的进程，也可以创建当前进程的子进程。

在linux内核空间，每个进程都有一个独立的数据结构，用来保存该进程的ID、优先级、地址的空间等信息，这个结构也被称做进程控制块（Process Control Block）。所谓的进程管理就是对进程控制块的管理。

linux的进程是通过fork()函数系统调用产生的。调用fork()的进程叫做父进程，生成的进程叫做子进程。子进程被创建的时候，除了进程ID外，其它数据结构与父进程完全一致。在fork()系统调用创建内存之后，子进程马上被加入内核的进程调试队列，然后使用exec()系统调用，把程序的代码加入到子进程的地址空间，之后子进程就开始执行自己的代码。

在一个系统上可以有多个进程，但是一般情况下只有一个CPU，在同一个时刻只能有一个进程在工作，即使有多个CPU，也不可能和进程的数量一样多。如果让若干的进程都能在CPU上工作，这就是进程管理子系统的工作。linux内核设计了存放进程队列的结构，在一个系统上会有若干队列，分别存放不同状态的进程。一个进程可以有若干状态，具体是由操作系统来定义的，但是至少包含运行态、就绪态和等待3种状态，内核设计了对应的队列存放对应状态的进程控制块。

当一个用户进程被加载后，会进入就绪态，被加入到就绪态队列，CPU时间被轮转到就绪态队列后，切换到进程的代码，进程被执行，当进程的时间片到了以后被换出。如果进程发生I/O操作也会被提前被换出，并且存放到等待队列，当I/O请求返回后，进程又被放入就绪队列。linux系统对进程队列的管理设计了若干不同的方法，主要的目的是提高进程调试的稳定性。

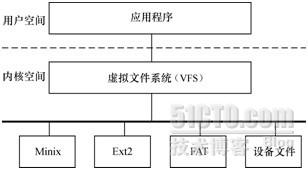
2．内存管理

内存管理的主要作用是控制多个进程安全地共享主内存区域。当CPU提供内存管理单元（MMU）时，Linux内存管理完成为每个进程进行虚拟内存到物理内存的转换。Linux 2.6引入了对无MMU CPU的支持。

如下图3所示，一般而言，Linux的每个进程享有4GB的内存空间，0～3GB属于用户空间，3～4GB属于内核空间，内核空间对常规内存、I/O设备内存以及高端内存存在不同的处理方式。

使用虚拟内存技术的计算机，内存管理的硬件按照分页方式管理内存。分页方式是把计算机系统的物理内存按照相同大小等分，每个内存分片称作内存页，通常内存页大小是4KB。Linux内核的内存管理子系统管理虚拟内存与物理内存之间的映射关系，以及系统可用内存空间。内存管理要管理的不仅是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。

在支持多用户的系统上，由于内存占用的增大，容易出现物理内存被消耗尽的情况。为了解决物理内存被耗尽的问题，内存管理子系统规定页面可以移出内存并放入磁盘中，这个过程称为交换。内存管理的源代码可以在./linux/mm中找到。



3．虚拟文件系统

如下图4所示，Linux虚拟文件系统（VFS）隐藏各种了硬件的具体细节，为所有的设备提供了统一的接口。而且，它独立于各个具体的文件系统，是对各种文件系统的一个抽象，它使用超级块super block存放文件系统相关信息，使用索引节点inode存放文件的物理信息，使用目录项dentry存放文件的逻辑信息。

在不同格式的文件分区上，程序都可以正确地读写文件，并且结果是一样的。有时在使用linux系统的时候发现，可以在不同类型的文件分区内直接复制文件，对应用程序来说，并不知道文件系统的类型，甚至不知道文件的类型，这就是虚拟文件系统在背后做的工作。虚拟文件系统屏蔽了不同文件系统间的差异，向用户提供了统一的接口。

虚拟文件系统，即VFS（Virtual File System）是Linux内核中的一个软件抽象层。它通过一些数据结构及其方法向实际的文件系统如ext2，vfat等提供接口机制。通过使用同一套文件 I/O 系统调用即可对Linux中的任意文件进行操作而无需考虑其所在的具体文件系统格式；更进一步，文件操作可以在不同文件系统之间进行。在linux系统中，一切都可以被看做是文件。不仅普通的文本文件、目录可以当做文件进行处理，而且字符设备、块设备、套接字等都可以被当做文件进行处理。这些文件虽然类型不同，但是却使用同一种操作方法。这也是UNIX/Linux设计的基本哲学之一。

虚拟文件系统(简称VFS)是实现“一切都是文件”特性的关键，是Linux内核的一个软件层，向用户空间的程序提供文件系统接口；同时提供了内核中的一个抽象功能，允许不同类型的文件系统存在。VFS可以被理解为一种抽象的接口标准，系统中所有的文件系统不仅依靠VFS共存，也依靠VFS协同工作。为了能够支持不同的文件系统，VFS定义了所有文件系统都支持的、最基本的一个概念上的接口和数据结构，在实现一个具体的文件系统的时候，需要向VFS提供符合VFS标准的接口和数据结构，不同的文件系统可能在实体概念上有差别，但是使用VFS接口时需要和VFS定义的概念保持一致，只有这样，才能实现对用户的文件系统无关性。VFS隐藏了具体文件系统的操作细节，所以，在VFS这一层以及内核其他部分看来，所有的文件系统都是相同的。对文件系统访问的系统调用通过VFS软件层处理，VFS根据访问的请求调用不同的文件系统驱动的函数处理用户的请求。文件系统的代码在访问物理设备的时候，需要使用物理设备驱动访问真正的硬件。

4．网络接口

网络接口提供了对各种网络标准的存取和各种网络硬件的支持。如下图5所示，在Linux中网络接口可分为网络协议和网络驱动程序，网络协议部分负责实现每一种可能的网络传输协议，网络设备驱动程序负责与硬件设备通信，每一种可能的硬件设备都有相应的设备驱动程序。

写网络应用程序，使用socket通过TCP/IP协议与其他机器通信，和前面介绍的内核子系统相似，socket相关的函数也是通过内核的子系统完成的，担当这部分任务的是内核的网络子系统，有时也把这部分代码称为“网络堆栈”。Linux内核提供了优秀的网络处理能力和功能，这与网络堆栈代码的设计思想是分不开的，Linux的网络堆栈部分沿袭了传统的层次结构，网络数据从用户进程到达实际的网络设备需要四个层次：用户进程，套接字，网络协议，网络设备。

实际上，在每层里面还可以分为好多层次，数据传输的路径是按照层次来的，不能跨越某个层次。linux网络子系统对网络层次采用了类似面向对象的设计思路，把需要处理的层次抽象为不同的实体，并且定义了实体之间的关系和数据处理流程：

（1）网络协议：网络协议可以理解为一种语言，用于网络中不同设备之间的通信，是一种通信的规范。

（2）套接字：套接字是内核与用户程序的接口，一个套接字对应一个数据连接，并且向用户提供了文件I/O，用户可以像操作文件一样在数据连接上收发数据，具体的协议处理由网络协议部分处理。套接字是用户使用网络的接口。

（3）设备接口：设备接口是网络子系统中软件和硬件的接口，用户的数据最终是需要通过网络硬件设备发送和接收的，网络设备千差万别，设备驱动也不尽相同，通过设备接口屏蔽了具体设备驱动的差异。

（4）网络缓冲区：网络缓冲区也称为套接字缓冲区（sk\_buff），是网络子系统中的一个重要结构。网络传输数据存在许多不定因素，除了物理设备对传输数据的限制（例如MMU），网络受到干扰、丢包、重传等，都会造成数据的不稳定，网络缓冲区通过对网络数据的重新整理，使业务处理的数据包是完整的。网络缓冲区是内存中的一块缓冲区，是网络系统与内存管理的接口。

5．进程通信

进程通信支持提供进程之间的通信，Linux支持进程间的多种通信机制，包含信号量、共享内存、管道等，这些机制可协助多个进程、多资源的互斥访问、进程间的同步和消息传递。

子系统之间的依赖关系：

Linux内核的5个组成部分之间的依赖关系如下：

·进程调度与内存管理之间的关系：这两个子系统互相依赖。在多道程序环境下，程序要运行必须为之创建进程，而创建进程的第一件事情，就是将程序和数据装入内存。

·进程间通信与内存管理的关系：进程间通信子系统要依赖内存管理支持共享内存通信机制，这种机制允许两个进程除了拥有自己的私有空间，还可以存取共同的内存区域。

·虚拟文件系统与网络接口之间的关系：虚拟文件系统利用网络接口支持网络文件系统(NFS)，也利用内存管理支持RAMDISK设备。

·内存管理与虚拟文件系统之间的关系：内存管理利用虚拟文件系统支持交换，交换进程（swapd）定期由调度程序调度，这也是内存管理依赖于进程调度的惟一原因。当一个进程存取的内存映射被换出时，内存管理向文件系统发出请求，同时，挂起当前正在运行的进程。

除了这些依赖关系外，内核中的所有子系统还要依赖于一些共同的资源。这些资源包括所有子系统都用到的例程，如分配和释放内存空间的函数、打印警告或错误信息的函数及系统提供的调试例程等。