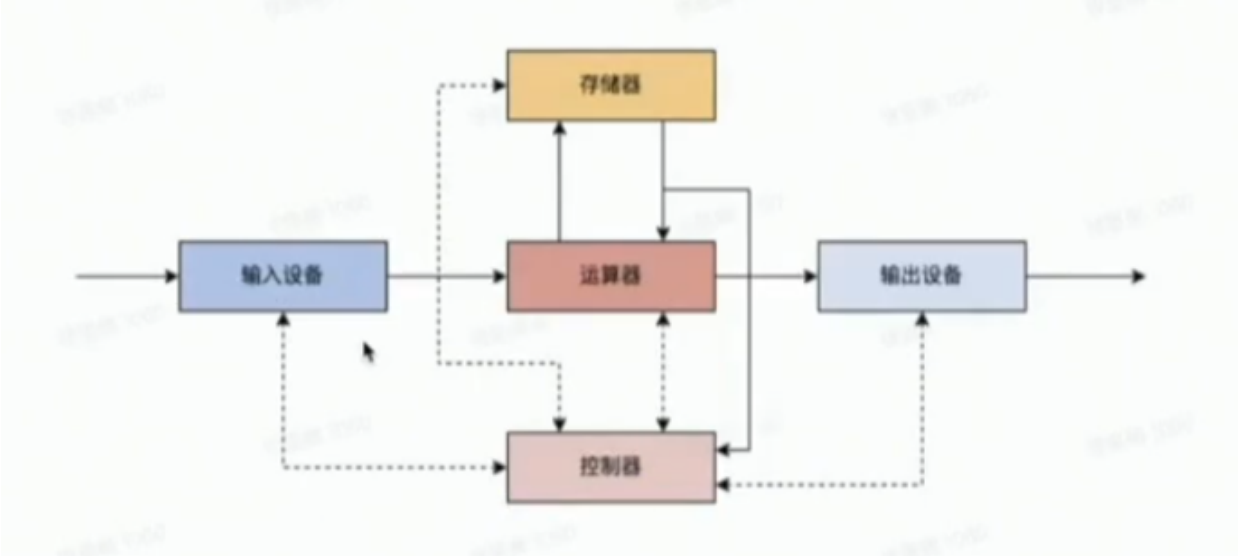
Linux基础

学习Linux 的价值

- Linux是现代化应用程序交付的首选平台，无论是部署在裸机、虚拟化还是容器化环境
- 公司内部服务（TCE、FaaS、SCM）统一使用DebianLinux系统
- 熟悉Linux基础指令，熟练运维前端常用服务 （Nginx，Node.js）
- 加深对操作系统概念和实现的理解，夯实基础知识

计算机硬件

计算机五大基本单元



- 控制器

> - 控制器是计算机的“大脑”，用于控制计算机中的各种操作。它接收指令，解码指令，调度指令，并且通过总线将指令发送到其他单元，以控制它们执行指令
> - 举例：计算机执行一个打印操作时，控制器会通过运算器进行相关运算，然后将需要打印的数据存储到存储器单元中，最后通过输出单元将数据输出到打印机中

- 运算器

> - 运算器是计算机中的算术和逻辑单元，用于执行各种算术和逻辑运算。它由ALU（算术逻辑单元）和其他寄存器组成
> - 举例：计算机执行加法操作时，将需要计算的两个数存储在寄存器中，运算器会从寄存器中读取这两个数并进行加法运算，将结果存储到另一个寄存器中

- 存储器单元

> - 存储器单元是计算机中的存储单元，用于存储程序和数据。它分为内存和外存两部分，内存一般指主存储器，外存一般指磁盘等外部存储设备
> - 举例：主存储器中存储着当前正在执行的程序和需要处理的数据，而辅助存储器则用于长期存储数据和程序

- 输入单元

> - 输入单元是计算机中的输入设备，用于接收外部数据并将其传输到计算机系统中。例如，键盘、鼠标、扫描仪等都是输入单元
> - 举例：键盘就是一种输入单元，可以将输入的字符或指令送到计算机中进行处理

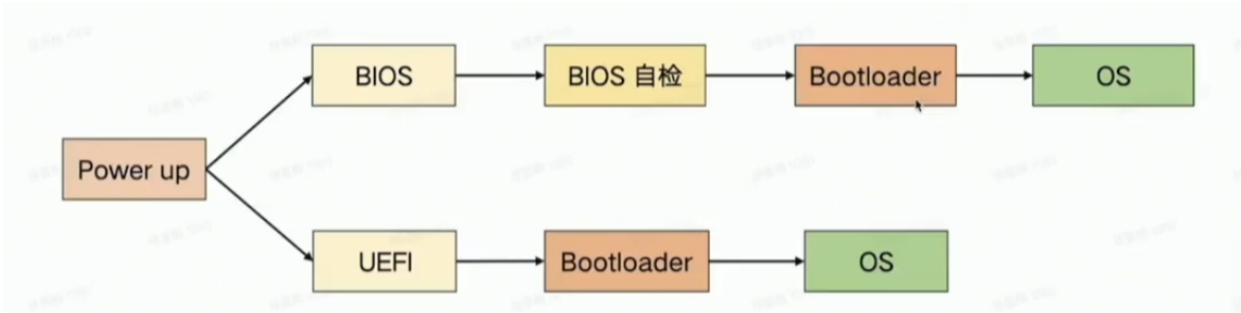
- 输出单元

> - 输出单元是计算机中的输出设备，用于将计算机系统中的数据传输到外部环境中。例如，显示器、打印机、喇叭等都是输出单元
> - 举例：显示器、打印机等就是一种输出单元，可以将计算机处理后的数据显示出来或者打印出来

计算机操作系统

> 管理和控制计算机系统中的硬件和软件资源，用于在用户与系统硬件之间传递信息。
>
> - 承上启下：
>   - 承上：在操作系统之上可以运用我们的计算机应用程序
>   - 启下：可以直接与硬件做出交互

- 问题思考：
- **程序启动必须有操作系统来执行，那操作系统本身也是一个程序，那是如何在开机时被执行的呢?**



> 在计算机启动时，最先被执行的是计算机 BIOS（Basic Input/Output System）或 UEFI（Unified Extensible Firmware Interface），它们是计算机的固件，也就是硬件上的软件。BIOS 或 UEFI 将会执行 POST（Power On Self Test，自检程序）以确认硬件设备是否正常。
>
> 接着，BIOS 或 UEFI 会寻找启动盘（通常是硬盘或者 USB 设备），读取启动盘中的引导程序。引导程序是一个小程序，它被放置在启动盘的特定位置，用于启动操作系统。

> 当引导程序被加载后，它会加载操作系统内核和初始化程序，并将控制权转移到操作系统内核中，启动操作系统的运行。操作系统会根据用户或者系统设置，加载相应的服务和应用程序，提供计算机的各种功能

BIOS与UEFI

> BIOS和UEFI都是计算机的固件，也就是硬件上的软件。它们的作用是在计算机启动时初始化硬件，检测设备是否正常，然后启动操作系统。
>
> **BIOS**（Basic Input/Output System，基本输入输出系统）是一种早期的固件，它在计算机启动时负责执行POST（Power On Self Test，自检程序），检测硬件设备是否正常，然后加载引导程序，启动操作系统。BIOS存储在主板上的闪存芯片中，由于其限制比较多，如容量小、功能简单、启动速度慢等，已逐渐被新一代的UEFI所取代。
>
> **UEFI**（Unified Extensible Firmware Interface，统一可扩展固件接口）是BIOS的后继者，是一种新型的计算机固件，提供比BIOS更多的功能和扩展性。UEFI支持更大的启动盘和更多的文件系统，也支持更高级的安全和启动选项，同时启动速度更快。UEFI通常存储在主板上的闪存芯片中，并由厂商提供升级固件的方式，使其支持新的硬件和功能。
>
> 在操作系统安装时，需要选择与BIOS或UEFI兼容的启动方式。在BIOS时代，常用的启动方式是Legacy BIOS（传统BIOS）启动模式，而在UEFI时代，常用的启动方式是UEFI启动模式。通常情况下，UEFI启动方式更为推荐，因为它提供了更多的功能和扩展性，同时也支持传统BIOS启动方式，以兼容老的硬件设备。

流程图	含义
BIOS自检（Basic Input/Output System Self-Test）	BIOS自检是计算机开机时自动运行的硬件诊断程序，它负责检查计算机硬件是否正常工作。自检过程包括以下内容：检查CPU、内存、磁盘驱动器、键盘、鼠标、打印机等硬件设备，以及系统时钟、电池电量等。如果自检过程出现错误，计算机将会发出一系列的声音和/或显示信息，提示用户出现了哪些问题
Bootloader（引导程序）	Bootloader是计算机启动过程中第一个被执行的程序，它负责在计算机启动时加载操作系统。当计算机启动时，BIOS或UEFI会寻找可启动设备，并将控制权交给该设备上的引导程序。引导程序会在可引导设备上查找操作系统的引导记录，并将控制权传递给该记录。如果找到引导记录，则引导程序会将控制权交给该记录，从而启动操作系统
操作系统OS	略

Linux系统概览

Linux发展简史

1. 1969年，Uniix诞生于贝尔实验室
2. 1984年，贝尔实验室将Unix商业化
3. 1984年，Tanenbaum开发Minix操作系统用于教学并开放源码
4. 1984年，Richard M.Stallman发起自由软件(FSF)与GNU项目，起草GPL（通用公共许可）协议

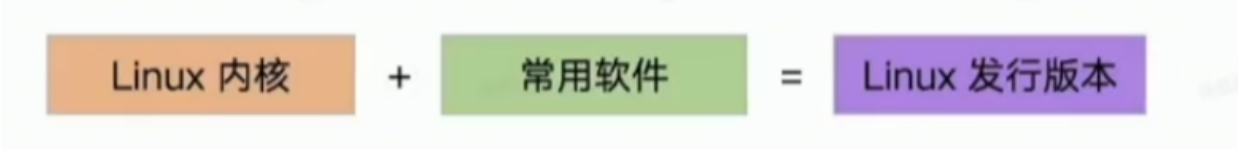
5. 1991年，Linus Torvalds:受Minix影响实现初版的Linux内核
6. 1992年，Liux内核以GPL协议发行V1.0

- 以上都处于诞生和初期的发展

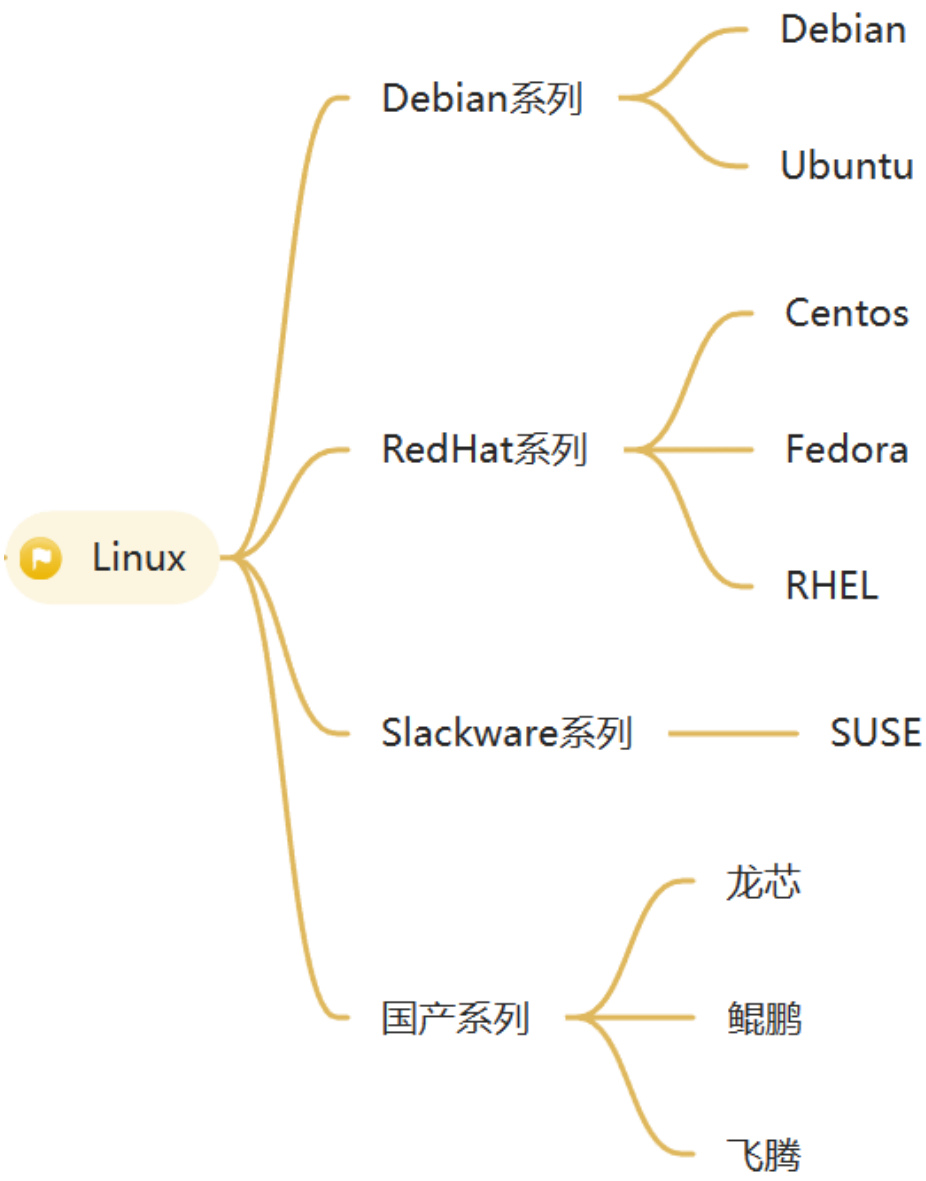
> 1. 1991年~1994年：诞生和初期发展
>
> 1991年，Linus Torvalds在芬兰大学创建了Linux项目，最初只是一个小型的自由软件内核项目。Linus的目标是开发一个类Unix的操作系统，能在普通的PC上运行。最初的Linux内核只有几千行代码，但随着时间的推移，越来越多的程序员加入了开发队伍，使得Linux内核逐渐变得庞大和复杂。
>
> 2. 1995年~2005年：商业化和扩张期
>
> 在这个时期，Linux发展迅速，开始获得商业公司的广泛认可。1995年，Red Hat公司成立，开始提供基于Linux的商业服务和支持，从而打响了Linux商业化的先声。其他的公司如SUSE和Debian也相继成立，并开始提供商业化的Linux发行版。同时，Linux社区也迅速扩张，越来越多的开发者加入其中，贡献代码和支持。
>
> 3. 2005年~至今：成熟期和广泛应用
>
> 随着时间的推移，Linux内核逐渐成熟，并被广泛应用于服务器、嵌入式系统、移动设备和个人电脑等领域。目前，Linux已经成为世界上最流行的操作系统之一，应用领域涉及云计算、人工智能、物联网、区块链等多个领域。此外，Linux社区也继续发展壮大，新的发行版、工具和技术层出不穷，使得Linux在未来的发展中仍然具有广泛的应用前景。

Linux版本

- 内核版本
- 发行版本



- **主流的Linux版本分支**



查看Linux系统内核版本

```
#方法1
uname -a
//显示系统的所有信息，包括内核版本号、操作系统发行版、主机名、处理器类型和架构等等
uname -r
//只显示当前Linux系统的内核版本号

#方法2
cat /proc/version
//cat /proc/version 命令用于显示当前 Linux 系统内核的版本号、编译者和编译日期。该命令读取 /proc/version 文件的内容并将其输出到终端上。

//该命令可以提供有关 Linux 内核的基本信息，包括内核版本、内核编译器和内核构建日期等。此外，它还提供了有关操作系统的其他信息，例如 GNU 工具链的版本和 CPU 架构信息。

//一些 Linux 系统中可能没有 /proc/version 文件，但是它们通常会提供其他方式来查看内核版本号，例如方法1的 uname -r 命令
```

Linux系统应用领域

- IT服务器(操作系统、虚拟化和云计算)
- 嵌入式和智能设备
- 个人办公桌面
- 学术研究与软件研发

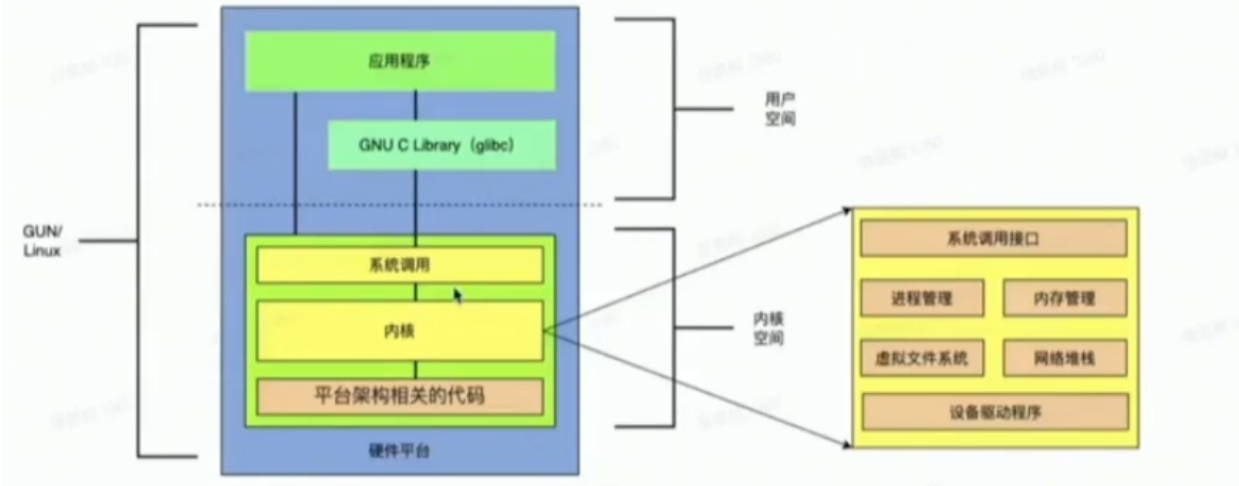
Linux系统结构

Linux四个主要部分

- 内核
  - Linux操作系统的核心部分，它管理计算机硬件的资源，包括CPU、内存、磁盘、网络等。它是操作系统与硬件之间的接口，提供了一个操作系统所需的各种基本服务和功能，如进程管理、文件系统、内存管理、网络协议栈等。
- shell
  - Shell是用户与Linux内核之间的接口，是一个命令解释器，提供了一种命令行界面供用户进行交互。在Shell中，用户可以输入命令和参数，执行脚本和程序，管理文件和目录等
- 文件系统
  - 文件系统是Linux操作系统中用于管理文件和目录的机制，是用户与操作系统之间进行文件交互的接口。Linux支持多种文件系统，如ext4、NTFS、FAT32等。它们管理着磁盘上的文件和目录，并提供文件读写、权限控制等基本功能
- 应用程序
  - 应用程序是运行在Linux操作系统上的各种软件，如文本编辑器、浏览器、视频播放器、编译器等。Linux操作系统拥有众多的应用程序，涵盖了各种领域，可以满足不同用户的需求

Linux体系结构

> Linux 的体系结构是一个多层次的体系结构，包括硬件层、内核层、应用程序层和用户层。



Linux 的体系结构	含义
硬件层	包括计算机硬件设备，如 CPU、内存、磁盘、网络等
内核层	是 Linux 的核心，负责管理硬件和提供系统服务。它包括系统调用、设备驱动程序、网络协议栈、虚拟文件系统等
应用程序层	是构建在内核之上的各种应用程序，如图形用户界面、Web 服务器、数据库、邮件服务器等
用户层	是用户与系统交互的界面，包括 shell、图形用户界面、命令行工具等。用户可以通过这些界面来与系统交互、操作文件和程序

用户空间和内核空间

- 用户空间是指应用程序运行的空间，包括用户应用程序、库、各种进程、服务等。用户空间的特点是受限制的权限，应用程序只能访问自己所拥有的资源，如自己的进程空间、自己的内存、自己的磁盘空间等。同时，用户空间还包括了各种 shell，用户可以通过 shell 进行命令行操作，从而控制整个系统
- 内核空间是指操作系统内核运行的空间，包括各种驱动程序、系统调用等。内核空间的特点是拥有系统级的权限，可以访问所有资源，包括 CPU、内存、磁盘、网络等。**内核空间的代码通常运行在特权级别最高的模式下**，可以直接操作硬件，因此内核空间的代码很少出错，一旦出错会导致整个系统崩溃
- 用户空间和内核空间之间通过系统调用进行通信。应用程序可以通过系统调用请求内核提供服务，如读写文件、网络通信、进程管理等。内核收到系统调用后会根据请求提供相应的服务，并返回结果给应用程序

> 中途涉及知识点：
>
> 应用程序发起 IO 请求的过程可以简单地概括为以下步骤：
>
> 1. 应用程序通过系统调用向内核发起 IO 请求。
> 2. 内核收到请求后，检查请求是否合法。如果请求不合法，则内核会向应用程序返回错误码。

3. 如果请求合法，内核会将请求加入到等待队列中，并将控制权返还给应用程序。
4. 当 IO 设备完成请求后，会产生一个中断信号通知内核。
5. 内核在中断处理程序中检查等待队列，将完成的请求从队列中移除，并将数据从内核空间复制到应用程序空间。如果有多个请求等待处理，内核会按照一定的策略进行调度，以确保公平性和效率。
6. 当请求处理完成后，内核会向应用程序发送一个信号，通知请求已经完成。
7. 应用程序收到信号后，继续执行自己的逻辑，处理已经完成的 IO 请求。

系统调用过程：

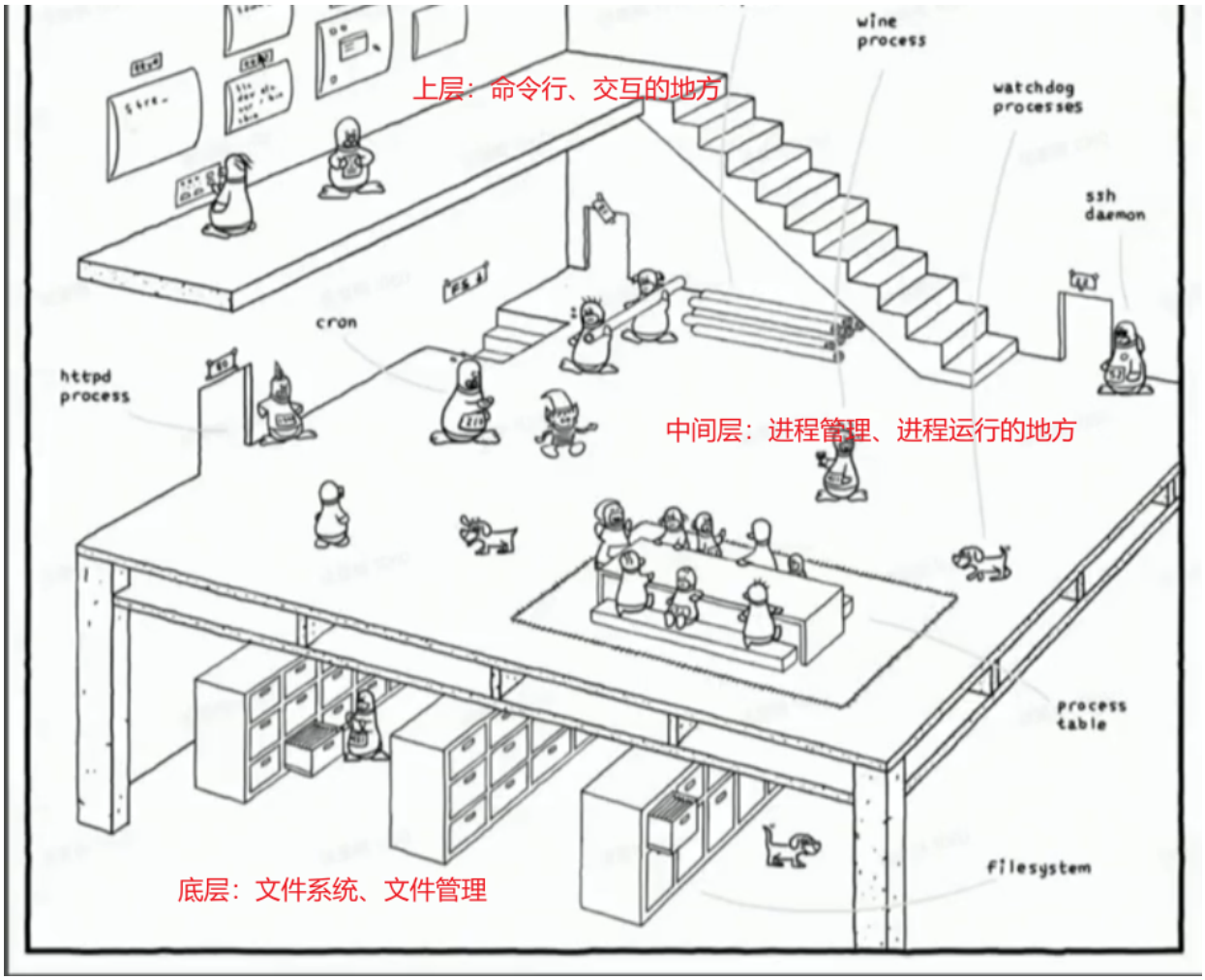
系统调用（system call）是操作系统向用户程序提供服务的接口，是操作系统的核心组成部分之一。它可以被视为用户程序与操作系统之间的桥梁，用户程序通过调用系统调用来向操作系统请求服务。

系统调用的一般过程如下：

1. 用户程序通过编写代码调用系统调用，指定调用的系统调用类型和参数。
2. 系统调用会将控制权转移到操作系统内核，即从用户态切换到内核态。
3. 在内核态中，操作系统会执行相应的系统调用，并根据调用的类型和参数进行相应的处理。
4. 处理完成后，操作系统将返回结果给用户程序，即从内核态切换回用户态。

在这个过程中，由于切换操作的开销比较大，因此系统调用的执行速度相对比较慢，因此在设计程序时应尽量避免频繁的系统调用操作。

- 内核是硬件与软件之间的中间层
- 内核是一个资源管理程序
- 内核提供一组面向系统的命令



- 像中间层靠着门的企鹅就是在监视着80端口
- 中间层两个肩扛管道的企鹅说明了，进程(企鹅)之间是允许进行管道通信的
- FS这个标志楼梯证明进程能够直接访问底层的

Linux系统结构 --进程管理

> 在Linux中，进程是指正在执行的程序实例。每个进程都拥有自己独立的虚拟地址空间、寄存器集合和打开文件的描述符等资源。进程是Linux中最为重要的概念之一

进程的特点

- 进程是正在执行的一个程序或命令
  - 进程是操作系统中正在执行的一个程序或命令的实例。每个进程都有一个唯一的进程标识符（PID）和一组相关的系统资源，例如内存、打开的文件和输入/输出设备
- 进程有自己的地址空间，占用一定的系统资源
  - 独立性：每个进程都是独立的实体，拥有自己的虚拟地址空间，因此一个进程无法访问另一个进程的内存空间，从而保证了进程的独立性和安全性
- 一个CPU核同一时间只能运行一个进程

    - 在单核 CPU 上，同时只能运行一个进程。因为 CPU 在同一时间只能执行一条指令，而每个进程都有自己的一组指令需要被执行，因此同一时间只能有一个进程在执行。当有多个进程需要执行时，操作系统会使用时间片轮转算法，轮流为每个进程分配 CPU 时间，以达到看起来多个进程同时运行的效果。但实际上，每个进程都只在短暂的时间内运行了一小段代码。在多核 CPU 上，可以同时运行多个进程，每个进程都可以被分配到一个 CPU 核心上运行。
- 进程由它的进程ID(PID)和它父进程的进程D(PPID)唯一识别
    - 程的唯一识别是通过进程ID（PID）来实现的，PID是一个唯一的数字标识符，用于区分正在运行的不同进程。在Linux中，每个进程都有一个唯一的PID，而且PID不会重复，因此可以通过PID来确定进程的身份
    - 与进程ID相关的另一个重要的属性是父进程ID（PPID），它是创建该进程的父进程的进程ID。在Linux中，每个进程都是由另一个进程创建的，所以每个进程都有一个PPID。通过PPID，我们可以建立进程之间的父子关系，形成进程树的结构
    - 除了PID和PPID之外，每个进程还有许多其他属性，例如进程的状态、优先级、打开的文件和共享内存等信息。这些属性可以通过/proc文件系统中的相应文件来查看

进程命令

- 查看启动的Nginx进程：
    - 可以使用 ps 命令查看正在运行的进程，配合 grep 命令可以过滤出含有关键字的进程

```
ps aux | grep nginx
```

- 查看某个进程：
    - 可以使用 ps 命令查看某个进程的信息

```
ps -p <pid>
top -p <pid> 命令查看指定进程的系统资源使用情况
```

- 关闭指定的进程：
    - 可以使用 kill 命令关闭指定进程。下面命令会向进程 ID 为 的进程发送终止信号，使其退出。

```
kill <pid>
```

- 全部进程动态实时视图：
    - 可以使用 top 命令查看所有进程的动态实时信息。会打印出一个实时更新的进程列表，包含 CPU 占用率、内存占用率等信息。可以使用快捷键 q 退出 top 命令

```
top
```

问题思考

系统中运行的程序远远大于CPU的核数，那Liux系统是如何实现同时运行这么多程序的?

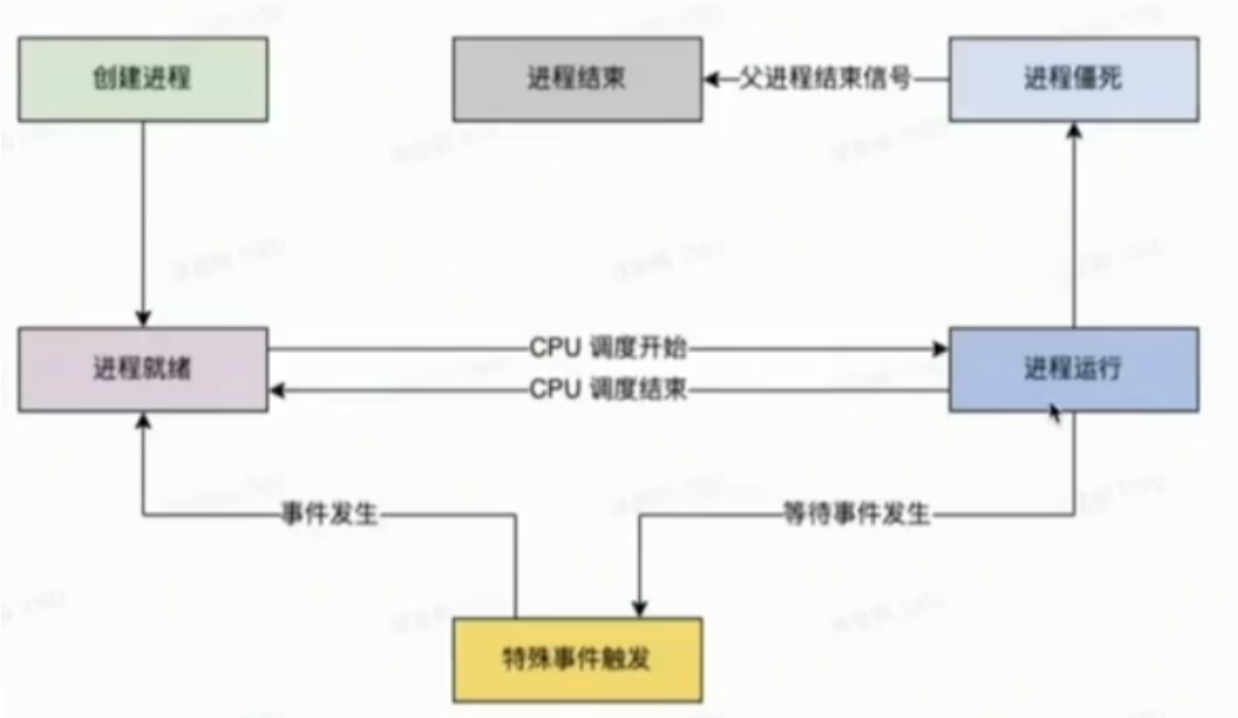
> **进程调度**是指操作系统按某种策略或规则选择进程占用CPU进行运行的过程：

- 它负责将系统中的进程分配到 CPU 的执行时间。因为一个系统中同时可能有很多进程需要运行，而 CPU 的时间是有限的，因此需要对这些进程进行调度，以便让它们都得到适当的时间片，从而使整个系统运行更加高效
- 进程调度的目标是提高系统的资源利用率和响应速度，通过有效地利用 CPU 时间，让系统可以同时运行多个进程，从而实现高效的资源共享。常见的进程调度算法包括先来先服务（FCFS）、短作业优先（SJF）、时间片轮转（RR）等

Linux系统通过操作系统内核对进程进行**调度**，以实现同时运行多个程序的目的。Linux内核采用了抢占式调度方式，每个进程都有自己的进程控制块（PCB），内核利用调度算法动态地将CPU时间片分配给各个进程，从而实现多任务处理。

当多个进程需要同时运行时，内核会对它们进行时间片轮转，即将CPU的使用时间按照时间片划分给多个进程，每个进程都能在一定时间内运行一段时间。当进程的时间片用完时，内核会将其挂起，将CPU时间片分配给下一个进程。

此外，Linux系统还采用了分时系统（Time Sharing System），即让每个进程都感觉到自己是独占整个系统的，因为系统在一段时间内快速地在各个进程之间切换，所以每个进程都有足够的响应速度和计算能力。这也是Linux系统能够同时运行多个程序的重要原因



- 创建阶段：进程在创建时，会分配资源并初始化进程控制块（Process Control Block，PCB），包括进程标识、程序计数器、CPU寄存器、内存分配情况等。
- 就绪阶段：进程在获得了运行所需的资源后，会被放置在就绪队列中等待CPU的分配。此时进程已经准备好运行了，只是还没有得到CPU的资源。
- 运行阶段：当进程被调度到CPU上运行时，进程的代码被加载到CPU中执行，这时进程进入运行状态。
- 阻塞阶段：在运行过程中，如果进程需要等待某些事件（如等待I/O操作完成），就会进入阻塞状态，此时进程会释放CPU资源，直到等待的事件完成。
- 结束阶段：当进程完成了它的任务或发生错误时，就会进入终止(僵死)状态。此时系统会回收该进程所占用的资源，并从进程表中删除该进程，也说明了这个进程结束了

进程的状态

> 其中，R、S、D、T是常见的进程状态。进程的状态会随着进程运行和系统调度而不断变化。进程状态可以通过命令ps -ef或top查看

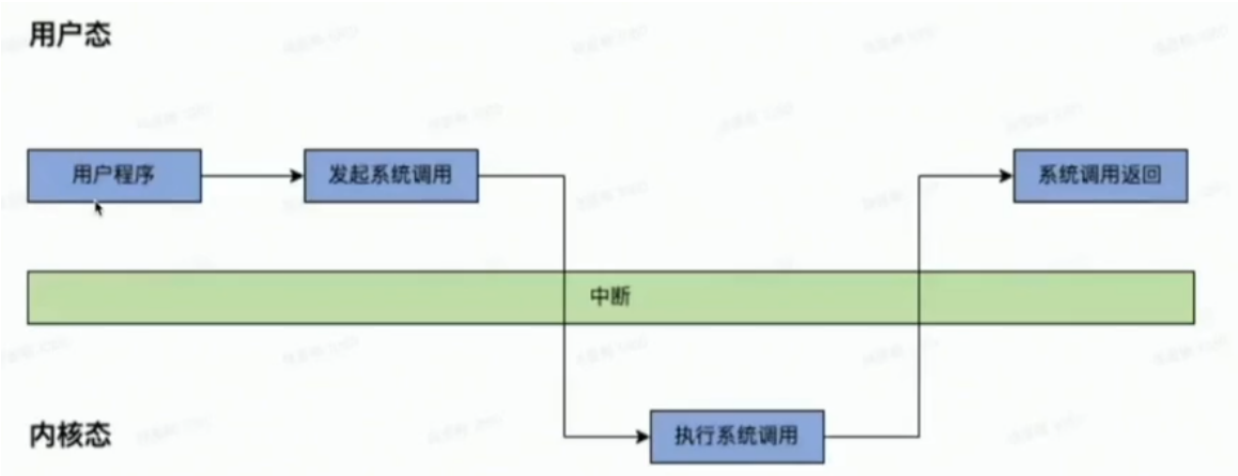
- R(TASK_RUNNING),可执行状态
- S(TASK_INTERRUPTIBLE),可中断的睡眠状态
- D(TASK_UNINTERRUPTIBLE),不可中断的睡眠状态
- T(TASK_STOPPED or TASK_TRACED),暂停状态或跟踪状态Z(TASK_DEAD-EXIT_ZOMBIE),退出状态，进程成为僵尸进程
- X(TASK_DEAD-EXIT_DEAD),退出状态，进程即将被销毁

进程调度原则

- 一个CPU核同一时间只能运行一个进程
    - 在多道程序设计中，CPU资源是必须被多个进程共享的。因为CPU核心数量是有限的，所以操作系统必须在多个进程之间进行调度，以便将CPU时间均匀地分配给它们。
- 每个进程有近乎相等的执行时间
    - 当进程开始执行时，操作系统会根据进程的优先级和时间片大小进行调度。在进程执行的过程中，操作系统会监测进程的执行时间，并在进程的时间片用尽之前将其挂起，以便为其他进程腾出CPU时间
- 对于逻辑CPU而言进程调度使用轮询的方式执行，当轮询完成则回到第一个进程反复
    - 操作系统通过轮询算法来进行进程调度。在轮询算法中，每个进程都被分配一个时间片，在时间片用尽之前，进程将一直运行。如果时间片用尽，则操作系统会将该进程挂起，并将CPU时间分配给下一个进程
- 进程执行消耗时间和进程量成正比
    - 进程的执行时间与进程数量成正比。当系统中有更多的进程需要执行时，操作系统需要更多的时间来进行进程调度。这就导致了更多的上下文切换和调度时间，进而降低了系统的性能

进程的系统调用

- 内核空间(Kernal Space):系统内核运行的空间
- 用户空间(User Space):应用程序运行的空间

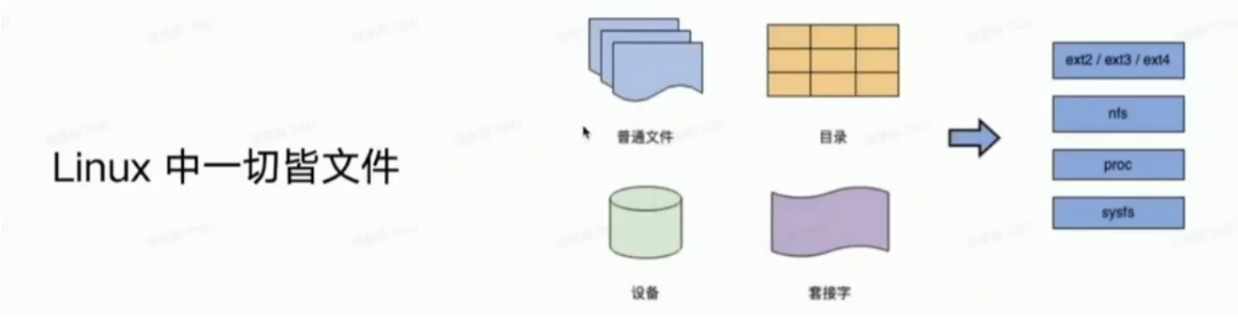


进程的系统调用是用户程序(用户态)与内核之间的一个接口，可以让用户程序获得内核提供的服务和功能。下面是进程的系统调用的基本流程：

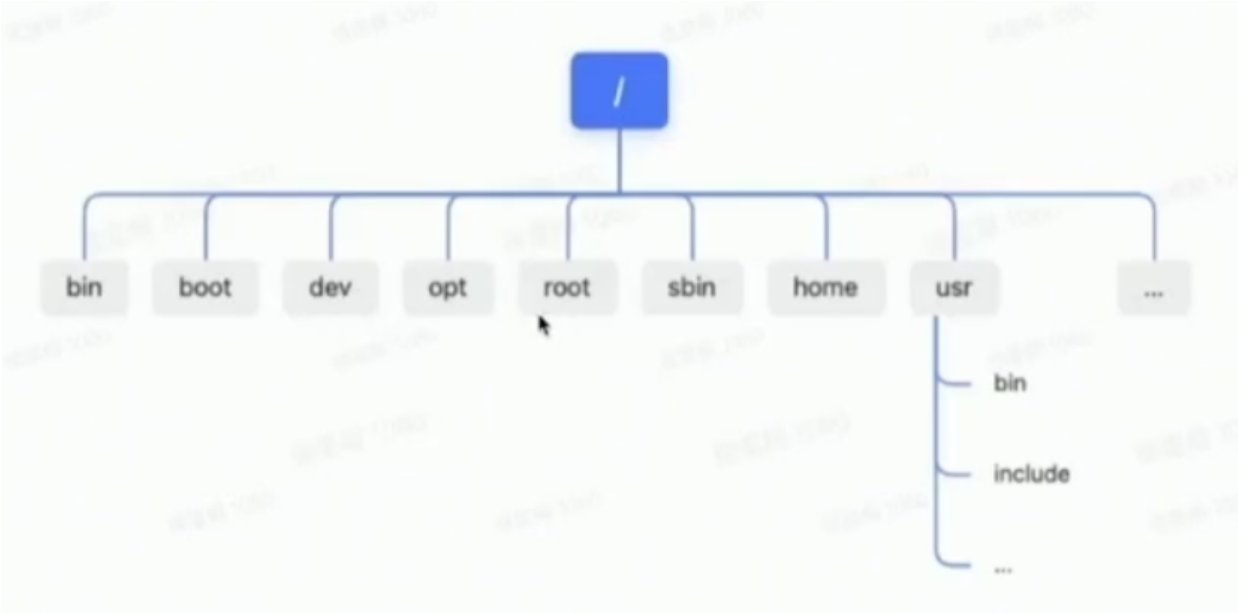
1. 用户程序发起系统调用，例如调用打开文件的系统调用 `open()`。
2. 程序经过用户态内核态切换后，进入内核态执行系统调用。
3. 内核执行系统调用，并返回结果给用户程序，例如返回一个文件句柄。
4. 程序再次经过内核态用户态切换后，回到用户态执行后续代码。

Linux --文件系统

- 文件系统是操作系统中负责管理持久数据的子系统，负责把用户的文件存到磁盘硬件中，持久化的保存文件。
    - 不同的文件有不同的类型



- Liux文件系统是采用树状的目录结构，
    - 最上层是 /（根）目录

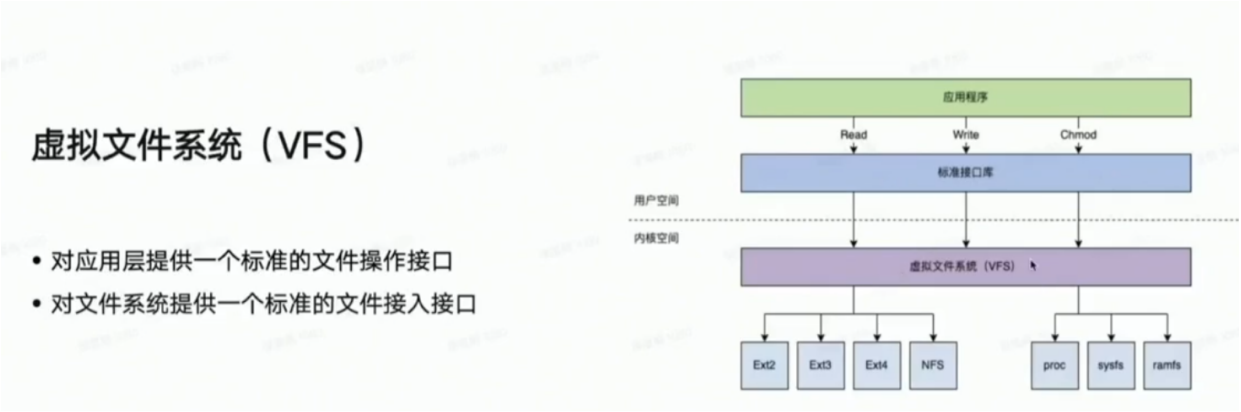


问题思考

Linux有这么多不同的文件系统，如何实现对用户提供统一调用接口的?

> Linux 为不同的文件系统提供了统一的调用接口，这就是虚拟文件系统（Virtual File System，简称 VFS）的概念。VFS 是 Linux 内核的核心模块之一，它提供了一套通用的文件系统接口，为应用程序提供了一种统一的、与底层文件系统无关的文件操作机制。

> 在 VFS 中，对所有的文件系统都采用统一的操作方式，这些操作会被映射到各个具体文件系统所对应的操作函数上。通过这种方式，VFS 能够屏蔽不同文件系统的差异性，为上层应用程序提供了一个统一的视图，使得应用程序不必关心文件的具体存储细节，而只需要关注文件操作本身。
>
> 因此，无论是 ext4、NTFS、FAT32 还是其他文件系统，应用程序都可以使用同样的方式来访问它们。这也是 Linux 系统在文件系统方面具有很强可扩展性和兼容性的一个重要原因。

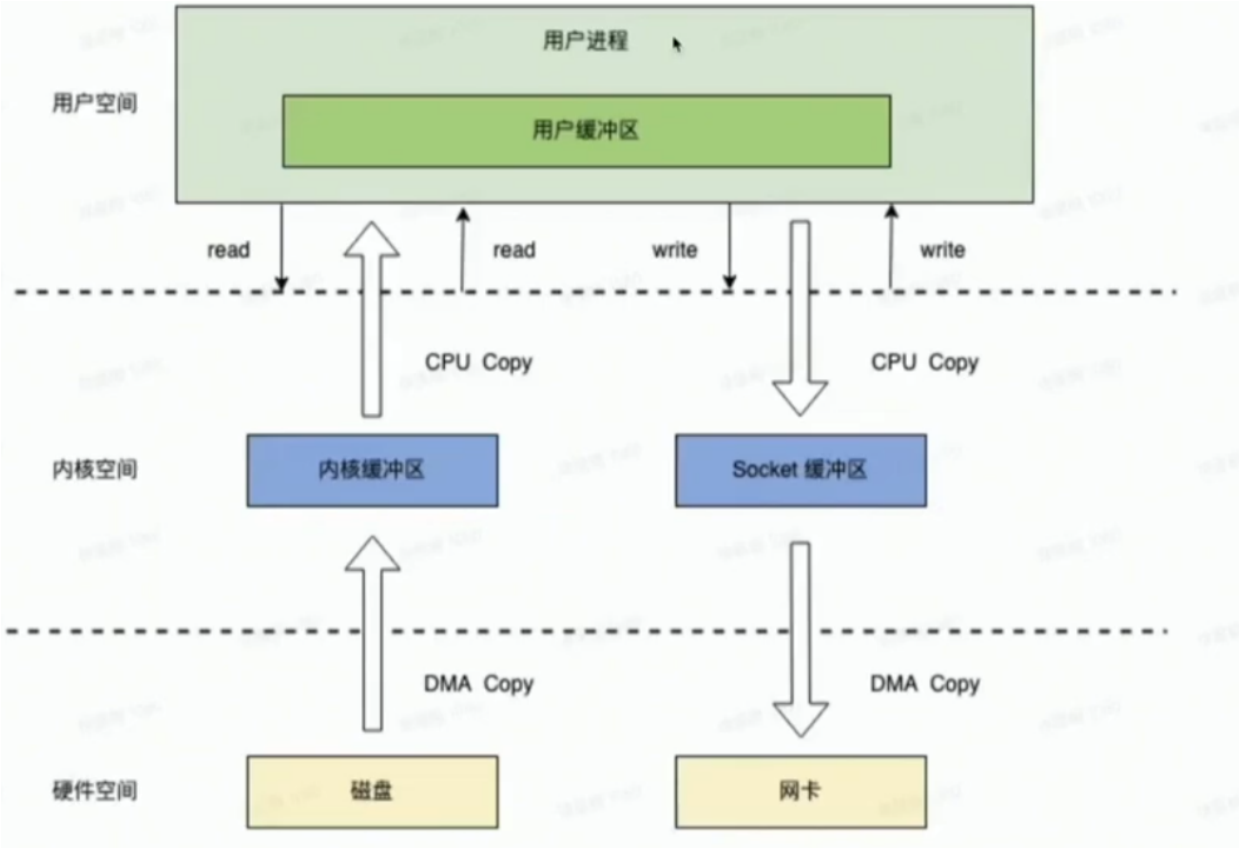



```
# df命令报告文件系统磁盘空间利用率
df -T

# mount 命令是挂载文件系统用的，不带任何参数运行，会打印包含文件系统类型在内的磁盘分区的信息
mount
```

简单文件操作命令

```
ls # 查看文件夹下内容

mkdir demo # 创建文件夹

mv demo /home # 移动demo 文件夹到 /home

rm -r demo # 删除demo文件夹

touch file.txt # 创建空文件

cp file.txt file_bak.txt #复制文件
```

文件读取流程



1. 用户空间

用户空间包括用户进程和用户缓冲区，用户进程通过系统调用发起文件读取请求，读取的数据存储在用户缓冲区中。

2. 内核空间

内核空间包括内核缓存区和Socket缓冲区，当用户进程发起文件读取请求后，操作系统内核会将文件内容从磁盘中读取到内核缓存区中，并将数据从内核缓存区复制到Socket缓冲区。

3. 硬件空间

硬件空间包括磁盘和网卡，当内核缓存区中的数据被复制到Socket缓冲区后，网卡会将数据发送到网络中，同时硬盘控制器从磁盘读取数据并传输到内核缓存区。

Linux -- 用户权限

- 用户账号
  - 普通用户账户：在系统中进行普通作业
  - 超级用户账户：在系统中对普通用户和整个系统进行管理
- 组账户
  - 标准组：可以容纳多个用户
  - 私有组：只有用户自己

```
# 查看当前登录用户信息
w
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
xxxx      pts/0    fdbd:ff1:ce00:11 14:57    2.00s  0.11s  0.00s w

# 查看当前用户所属的组
groups
xxxx tiger admin

# 查看用户的 uid 信息
id xxxx
uid=1001(xxxx) gid=1001(xxxx) groups=1001(xxxx),1000(tiger),2001(admin)
```

- 文件权限关于用户有三个概念：
  - 所有者：文件的所有者
  - 所在组：文件的所有者所在的组
  - 其他人：除文件所有者及所在组外的其他人
- **每个用户对于文件都有不同权限，包括读(R)、写(W)、执行(X)**

基础用户操作命令

- 在根目录创建一个文件夹，查看当前用户拥有文件夹的权限
  - 这是三个命令组合在一起执行的语句。
    1. `cd /` 切换到根目录
    2. `mkdir demo` 创建名为`demo`的文件夹
    3. `ls -ld demo` 查看`demo`文件夹的详细信息，包括文件夹的权限等。其中，`ls`是查看文件和目录的命令，`-l`选项是显示详细信息，`-d`选项是显示目录自身信息，而不是显示目录内文件信息。

```
cd  / && mkdir demo && ls -ld demo
```

- 创建一个用户，并赋予可写操作
    - sudo useradd 表示以管理员权限执行添加用户的操作，而 ceshi 则是指定要创建的用户的用户名。执行该命令后，系统会创建一个新的用户账户，并在系统中为其分配一个用户ID、主目录和默认shell等

```
sudo useradd ceshi
```

- 设置用户密码
    - 设置ceshi用户的密码的命令，其中sudo是用来获得超级用户权限，passwd是用来设置用户密码的命令，ceshi是指定要设置密码的用户

```
sudo passwd ceshi
```

- 切换 ceshi 用户登录
    - 在当前终端中切换到用户ceshi的身份。通过执行这个命令，你可以在终端中执行ceshi用户具有权限的操作

```
su ceshi
```

- 进入demo文件夹

```
cd demo
```

- 创建index.js文件，提示无权限，需要给ceshi用户demo文件夹的权限
    - 在当前目录下创建一个名为index.js的空文件的命令

```
touch index.js
```

- demo文件夹权限给ceshi用户
    - 将当前目录下的 `demo` 文件夹的所有文件和文件夹的拥有者(owner)和所属组(group)都修改为 `ceshi` 用户和 `ceshi` 组。其中 `-R` 参数表示递归修改

```
sudo chown -R ceshi:ceshi ./demo
```

- 切换ceshi 用户登录

```
su ceshi
```

- 进入demo文件夹

```
cd demo
```

- 创建index.js文件成功

```
touch index.js
```

Linux系统软件包管理器

- 软件包
  - 通常指的是一个应用程序，它可以是一个GUI应用程序、命令行工具或（其他软件程序需要的）软件库
- 软件包管理
  - 底层工具：主要用来处理安装和删除软件包文件等任务
    上层工具：主要用于数据的搜索任务和依赖解析任务

上层与底层工具的区别

底层工具主要用于底层的软件包管理操作，例如软件包的安装、卸载、更新等，其主要特点包括：

- 以命令行为主要界面；
- 操作灵活、功能强大；
- 操作相对复杂，需要较高的技术水平。

常见的底层工具包括：

- **DPKG**：Debian Linux 系统的底层软件包管理工具；
- **RPM**：Red Hat Linux 系统的底层软件包管理工具；
- yum：基于 RPM 的高级包管理器；
- apt：基于 dpkg 的高级包管理器。

操作系统	格式	软件包管理系统	前端工具
Debian	.deb	dpkg	apt, apt-get
Ubuntu	.deb	dpkg	apt, apt-get
CentOS	.rpm	rpm	yum
Fedora	.rpm	rpm	dnf
openSUSE	.rpm	rpm	zypper

上层工具则更加注重用户友好性，提供了一些图形化界面和便捷的操作方式，其主要特点包括：

- 提供图形化界面，操作简单方便；
- 可以方便地搜索、安装和卸载软件包；
- 功能相对有限。

常见的上层工具包括：

- Synaptic：适用于 Debian 系统的上层软件包管理工具；
- Yumex：适用于 Red Hat 系统的上层软件包管理工具；
- Apper：适用于 KDE 桌面环境的上层软件包管理工具；
- Gnome-Software：适用于 Gnome 桌面环境的上层软件包管理工具。

常用命令

```
列出所有可更新的软件清单命令：apt update
安装指定的软件命令：apt install<package_name>
安装多个软件包：apt install<package_1><package_2><package_3>
更新指定的软件命令：apt update<package_name>
删除软件包命令：apt remove<package_name>
查找软件包命令：apt search<keyword>
列出所有已安装的包：apt list-installed
```

管理软件源

> 就是类似我们平时使用npm的时候因为网络问题会切换的那个镜像源
>
> 镜像地址：https://mirrors.aliyun.com/