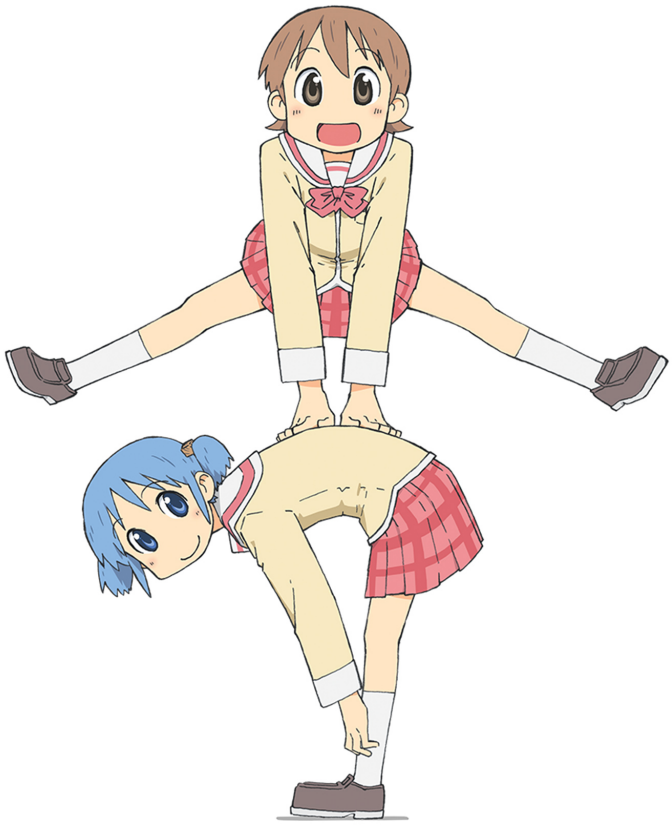


nichijou
my ordinary life

day01-系统-文件管理-系统命令

1.计算机组成原理

1.1 什么是计算机

- 计算机也称为 ”电脑“，电脑电脑：即通电的大脑；
    - 电脑二字其实蕴含了人类对计算机终极的期望；
    - 希望它能像人脑一样为我们工作，从而取代人力，并将人力解放出来；

1.2 为什么要有计算机

- 为什么要有计算机，或者人类为什么要造计算机?
    - 其实是为了执行人类的程序，从而将人力解放出来；（因为人存在很多不可控因素）
    - 所以计算机在造的时候，它每一部分的设计都是在模仿人的某个器官或功能去设计的；

1.3 计算机五大组成部分

- 计算机由五大组件组成，我们完全可以把计算机的五大组件比喻成人类的各种器官
    - 控制器
    - 运算器
    - 存储器
    - 输入设备 `Input/I` 设备
    - 输出设备 `Output/O` 设备

1.3.1 CPU

- 控制器：
    - 作用：是计算机的指挥系统，主要负责控制计算机其他所有组件如何工作的；
    - 比如：走路、跑、跳、说话都是谁在控制呢?
    - 类比：控制器-->人类大脑；

- 运算器：
  - 作用：运算及字面含义，主要包含数学运算、逻辑运算；
  - 比如：1+1=逻辑运算；上车看见好看姑娘，追还是不追=逻辑运算；
  - 类比：运算器-->人类大脑；
- 小结：
  - 其实控制器和运算器压根就不是两个硬件
  - 控制器+运算器其实就是 `CPU（芯片）` --> 人类的大脑（前脑+后脑）；

1.3.2 内存/硬盘

- 存储器/IO：
  - 作用：负责程序数据的存取；对于计算机来说，有了存储器，才有记忆功能
  - 分类：
    - 内存：内存基于电工作，通电就可以存储数据；
      - 优势：存取数据快；
      - 缺点：断电数据会丢失，仅能临时存储数据；
    - 外存：外置硬盘，基于磁工作；
      - 优势：断电数据不会丢失，可以永久保存数据；
      - 缺点：存取速度慢；
  - 类比：
    - 内存-->大脑的记忆功能（快、短期记忆）；
    - 硬盘-->随身携带小本本（慢、长期记忆）；
  - 举例：
    - 女朋友的生日假设是（0921），我们一般记忆在哪最合适（送命题）；
      - 1.可以记忆在大脑，快速响应，但如果搬砖敲你一下，失忆了怎么办；
      - 2.聪明的伙伴会说我记录在小本本上，忘了看一眼，然后延迟响应（那你可能..）；
    - 如果女朋友问她的生日是什么时间，怎么记忆最佳：
      - 第一步：单纯记录到脑子里可能会忘记，所以我将生日记录到小本本上；
      - 第二步：在女朋友每次询问我之前，将小本本拿出来看一眼，记到脑子中；
      - 第三步：在女朋友问我的时候，我直接从脑子记忆中提取她的生日日期即可；
  - 敲重点：如果程序的数据要快存快取：
    - 第一步：将应用程序存储至硬盘中，如果不用则存储下来即可；
    - 第二步：如果需要使用该程序，一定是需要将硬盘的数据加载到内存中；
    - 第三步：最后CPU读取内存中的指令，进行分析和处理；从而保证程序的执行速度；

1.3.3 输入设备

- 输入设备`Input`：
  - 作用：往计算机里面输入内容；（键盘、鼠标）
  - 比如：眼睛看、耳朵听；
  - 类比：输入设备-->人类的（眼、耳）；

1.3.4 输出设备

- 输出设备`Output`：
  - 作用：计算机向外输出数据的工具；（显示器、打印机）
  - 比如：人说话，人发布文章
  - 类比：输入设备-->`xx`；

1.3.5 五大组件总结

- 老师讲课，学生听课，老师是程序员，学生是计算机；（学生的器官都是计算机各部分组成）
  - 1.学生通过自己耳朵听、眼睛看，接收老师讲的知识；这个就是-->输入
  - 2.学生通过自己的神经、将接收的信息存入自己的短期记忆中；这个就是-->内存
  - 3.学生光听不行，还需要理解老师讲的知识，于是你的大脑从短期记忆里取出知识/指令，分析知识/指令，然后学习知识/执行指令 -->这就是cpu（取指、分析、执行）
  - 4.学生通过作业、给其他学生讲解、将学到的东西表达出来-->这就是输出
  - 5.学生想要永久将知识保存下来，进行长期记忆、需要将内容写到本子上；-->这就是硬盘

1.4 计算机三大核心硬件

- 我们将五大组成部分，进一步提炼出其中的三大核心硬件：（CPU、内存、磁盘）
  - 因为一个程序的运行与计算机三大核心硬件存在着特定的联系；
  - 前提：`人` --通过--> `语言` --控制--> `计算机（即人）`
  - 举例：我通过语言编写一段程序，控制计算机（人）做如下几件事：
    - 1.买烟；
    - 2.掏钱；
    - 3.回家；
  - 目的：控制人的身体去运转、替我们工作；
- 问题1：我们编写的程序没有详细描述他应该如何工作，那到底是计算机的哪个组件下发的控制指令；
  - 其实计算机的所有组件都受计算机的`CPU`控制；
  - 也就是程序是直接控制大脑`（CPU）`，由大脑`（CPU）`间接支配人的肉体（组件），从而实现程序支配肉体工作

- 问题2：如果我不想每次反复描述这件事，希望这个任务反复运行怎么办；
  - 计算机具备存储的就是内存和硬盘；
    - 如果直接存储在内存丢失了怎么办，难道在描述一次；所以这个程序是需要存储在硬盘上；
    - 也就是编写好的程序或者软件一定是存储在硬盘上的；
- 问题3：如果只有 `CPU` 和硬盘，能否将这段程序运行起来；
  - 其实是可以运行起来的， `CPU` 从硬盘中取出指令进行运行即可，但是存在问题；
    - `CPU` 的速度要远高于硬盘；如果每次都需要从硬盘数据中读取一条数据，然后CPU处理一条；然后继续读取、继续处理，一直反复这个过程，那么大量的时间都会浪费在数据的读取上；
  - 那我们该如何提升程序运行的速度呢，此时就需要内存的介入（人脑的记忆）；
    - 第一步：我们将要操作的步骤存储至磁盘（小本本）；
    - 第二步：将硬盘的数据加载进内存中（大脑的记忆）；
    - 第三步：CPU从内存中读取指令运行，效率非常高；

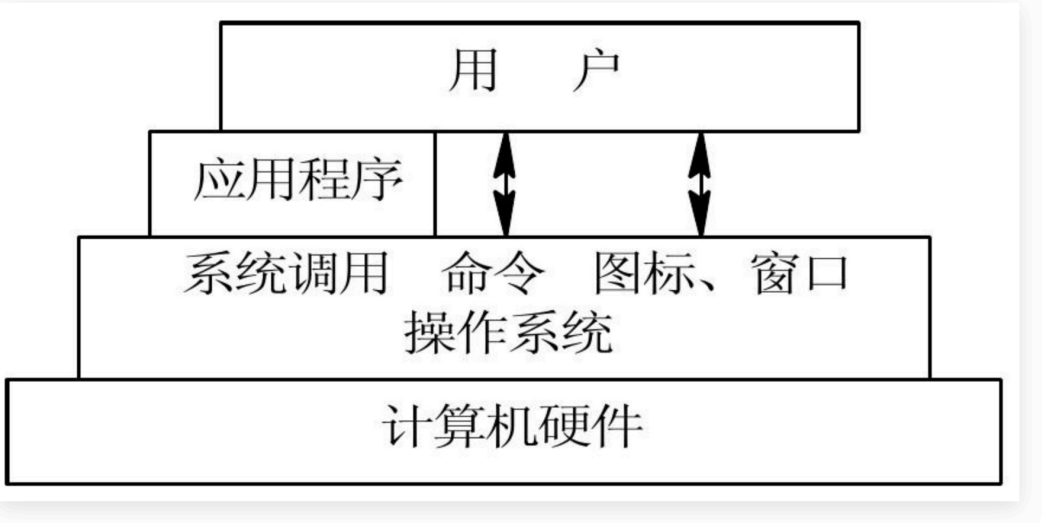
1.5 操作系统基本概念

1.5.1 操作系统由来

- 需求：
  - 开发一个编辑工具，该软件的一个核心业务就是文本编辑，编辑内容就牵扯到要操作计算机硬件；
- 问题：
  - 1.不管我们编写什么软件，最终的目的是为了控制硬件；
  - 2.但对于计算机而言，它是死的，它不可能自行运行，所有的硬件运行都需要软件进行支配；
- 实现：
  - 第一步：我们必须先开发一个 ”控制系统“ 来控制计算机的硬件基本运行；
  - 第二步：然后在开发编辑工具的业务功能，但凡涉及到要操作硬件，则调用控制系统；
  - 第三步：最后由控制系统来控制计算机硬件进行运行；

1.5.2 什么是操作系统

- 其实前面所所的控制系统有一个更好听的名称，操作系统 `Operation System, OS`
  - 1.操作系统是”应用软件“与”硬件“之间的一个桥梁；
  - 2.同时也是一个协调、管理、“控制计算机硬件资源”、“应用软件资源”的一个控制程序；



1.5.3 为什么需要操作系统

- 1.控制计算机的基本运行；
- 2.将硬件的复杂操作简单化，供上层应用程序使用；
- 3.为用户与计算机硬件之间提供 ”图形/命令行“ 工具；

1.2 Linux系统介绍

1.2.1 什么是Linux

- `Linux` 和我们常见的 `Windows` 一样，都是操作系统，但 `linux` 有两种含义；
  - 一种是 `Linus` 编写的开源操作系统的内核
  - 另一种是广义上的操作系统
- `Linux` 与 `Windows` 系统不同的是；
  - `Windows` 收费，不开源，主要用于日常办公、游戏、娱乐多一些。
  - `Linux` 免费，开源，主要用于服务器领域，性能稳定，安全。
- 例如: 淘宝、百度、腾讯等互联网公司，他们使用的服务器全都是 `Linux` 系统；

1.2.2 Linux发展历史

既然是历史，那就让他成为历史吧，因为我根本记不住历史。(因为我不是导游，不靠记历史赚钱)。
虽然历史不重要，但是还是需要了解 `Linux` 在发展过程中的一些重要人物

1.2.2.1 自动软件之父

- 自由软件之父 `Richard M. Stallman 1984` 发起了 `GNU` 组织

- `GUN`组织中有几个项目：
  - `copyleft`： 代表无版权。 `copyright`： 则代表有版权。
  - `opensource`： 开放源代码、软件谁都可以使用、谁都可以传播、谁都可以二次开发
  - `free`： 免费
  - `GPL`： 通用版权许可证协议，如果软件被打上GPL，那么任何人都可以对这个软件进行修改，但是修改完之后必须将源码发布出来，以便更好的传承下去。
- 总结：
  - `Linux` 中的软件百分之八十都是 `GPL` 提供；
  - 自由软件运动的口号是：“团结就是力量”；

1.2.2.2 Linux系统之父

- Linux 之父 `Linus Torvalds` 林纳斯.托瓦兹 `1991` 年 `Linux` 内核；
- 操作系统的核心称为“内核”，但内核并不就等于操作系统；
- 内核提供系统服务，比如文件管理、虚拟内存、设备I/O等；还包含一些基本的程序、编译器、shell等；所以单独的Linux内核没办法工作，须要有GNU项目的众多应用程序；
- 其实 `Linux` 官方叫法是 `GNU/Linux` 使用 `GNU` 的软件加上 `Linux` 内核，一般简称`Linux`

Linux 内核网站
linux 大神在2017-06-26来到中国

1.2.3 Linux系统发行版

我们现在说的 `Linux` 其实都是指的是发行版 `Distribution version`；就是使用 `Linux` 内核加上各种 `GNU` 的库文件、应用程序，构造而成的操作系统。
Linux发行版介绍`RHEL/Centos/Ubuntu/Suse`

- `Redhat` 企业级操作系统， `Linux` 的内核进行编译安装相应软件，进行专业的测试，然后进行发行；
- `CentOS` 社区企业级操作系统，改与 `Redhat` 完全开源；
- `Ubuntu` 社区维护， 现在主要做手机系统和电脑桌面系统；
- `Debian` 等等.....

1.2.4 为什么使用CentOS

- `CentOS` 是 `Community Enterprise Operating System` 的缩写
- 表示 "社区企业操作系统"
- `CentOS` 兼具 `Community` （社区）和 `Enterprise` （企业）的特性
- `CentOS` 稳定、长期支持（10年）大规模使用稳定；

1.3 虚拟化环境（Vmware）

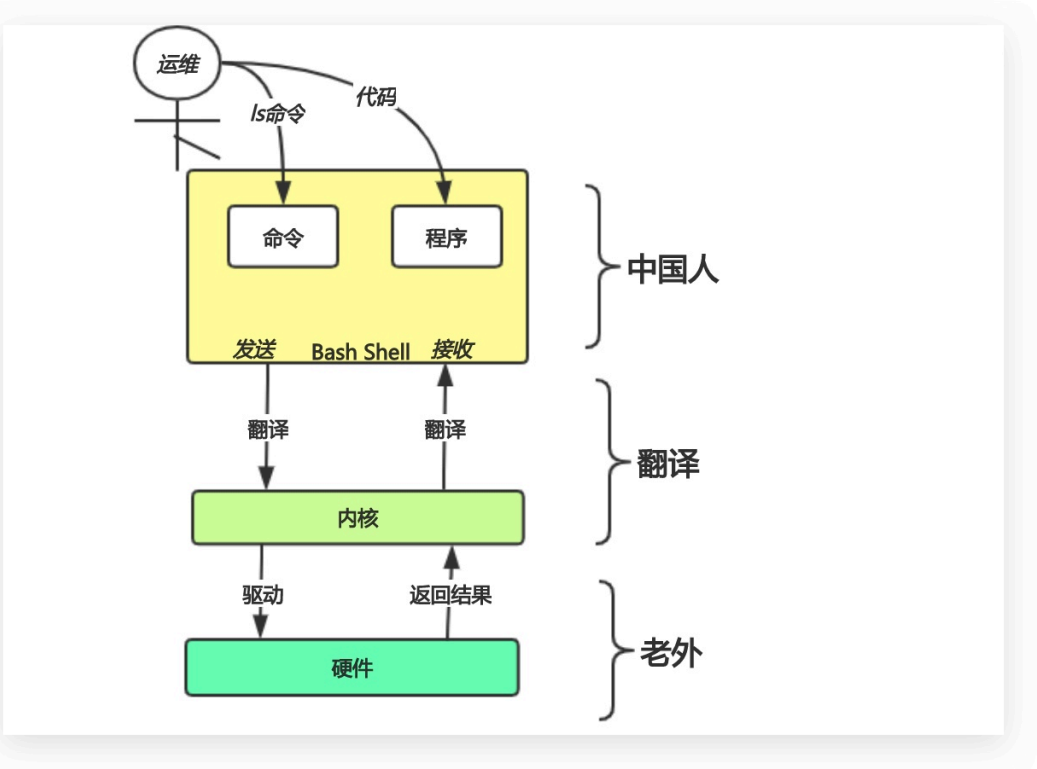
1.3.1 安装虚拟机

1.3.2 安装CentOS

1.4 人机交互接口Bash

1.4.1 什么是Bash shell

`Bash Shell` 是一个命令解释器，主要负责用户程序与内核进行交互操作的一种接口，将用户输入的命令翻译给内核，由内核驱动硬件，最终将处理后的结果输出至 `Bash Shell` 屏幕终端。



我们如何打开 `Bash Shell` 呢?
当我们使用远程连接工具连接 `linux` 服务，系统则会给打开一个默认的 `shell`，我们可在这个界面执行命令、比如：获取系统当前时间，创建一个用户等等

1.4.2 Bash Shell能干什么

- 使用Shell实现对Linux系统的大部分管理，例如:
  - 1.文件管理
  - 2.权限管理
  - 3.用户管理
  - 4.磁盘管理

- 5.网络管理
- 6.软件管理
- 7.服务管理
- 8.等等.

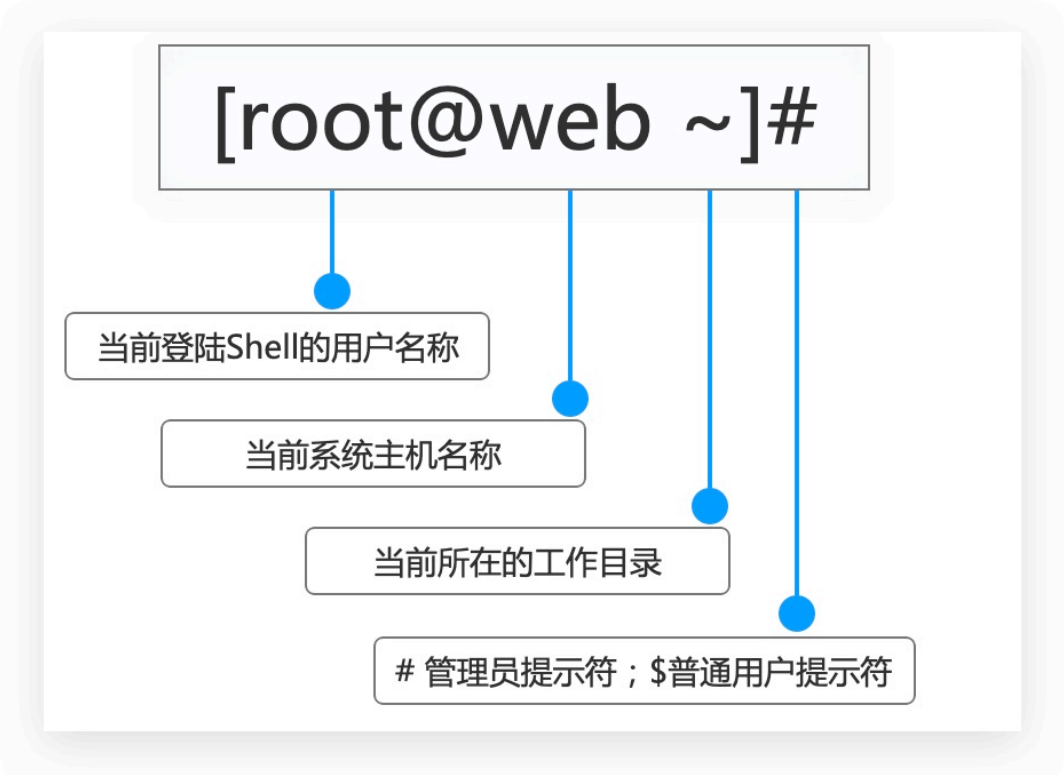
1.4.3 如何使用Bash Shell

- 单条命令--> 效率低 -->适合少量的工作
- shell脚本 --> 效率高-->适合重复性的工作
- 例如:创建100个用户，单纯输入命令需要执行100次，而如果使用Shell脚本则可以轻松解决；

```
[root@web ~]# cat useradd.sh
#!/usr/bin/bash
# 批量创建脚本
for i in {1..100}
do
    useradd alice-$i
    echo "alice-$i" is create ok..
done
```

1.4.4 Bash Shell提示符

当我们打开或者登陆到一个终端的时候都会显示一长串提示符 `[root@web ~]#`
提示符一般包含当前登陆的用户名 ， 主机名，以及当前工作路径等；



1.4.5 Bash Shell基础语法

`bash shell` 命令行，为用户提供命令输入，然后将执行结果返回给用户；

命令	选项	参数
command	options	parameters

命令示例如下：

```
[root@web ~]# ls              # 命令
[root@web ~]# ls -a           # 命令+选项
[root@web ~]# ls -a /home/    # 命令+选项+参数
```

- 命令：整条 `shell` 命令的主体功能
- 选项：用于调节命令的具体功能
  - 以 `-` 引导短格式选项（单个字符），例如 `-a`
  - 以 `--` 引导长格式选项（多个字符），例如 `--all`
  - 多个短格式选项可以写在一起，只用一个 `-` ，例如 `-al`
- 参数：命令操作的对象，如文件、目录名等
- 注意：命令必须开头，选项和参数位置可以发生变化

1.4.6 Bash Shell基本特性

1.4.6.1 补全功能tabs

- 1.命令补全：当忘记命令时，可以使用tabs进行补全；
- 2.目录补全：当需要查找文件目录层级比较多时，可以使用tabs快速补全，减少出错；

```
#查看ip时忘记具体了命令
[root@web ~]# ifcon
#按下tab键会自动补全
[root@web ~]# ifconfig

#按一下tab键没有反应，按两下tab键列出所有if开头的命令
[root@web ~]# if
if          ifconfig    ifenslave   ifrename
ifcfg       ifdown      ifnames     ifup

#linux目录较深，经常使用tab键进行补全，如果路径出错是没有办法补全
[root@web ~]# ls /etc/sysconfig/network-scripts/
```

1.4.6.2 常用快捷键ctrl

- 命令快捷键，快捷键可以帮助我们大大提升工作效率
  - `Ctrl + a`：光标跳转至正在输入的命令行的首部
  - `Ctrl + e`：光标跳转至正在输入的命令行的尾部
  - `Ctrl + c`：终止前台运行的程序
  - `Ctrl + d`：在shell中，ctrl-d表示推出当前shell。
  - `Ctrl + z`：将任务暂停，挂至后台
  - `Ctrl + l`：清屏，和clear命令等效。
  - `Ctrl + k`：删除从光标到行末的所有字符
  - `Ctrl + u`：删除从光标到行首的所有字符
  - `Ctrl + r`：搜索历史命令, 利用关键字

1.4.6.3 历史记录History

历史记录可用于追溯系统之前执行过什么命令，造成的故障；之前发生情况

1.使用双 `!!` 可执行上一条执行过的命令

```
[root@web ~]# ls
file.txt
[root@web ~]# !!
ls
file.txt
```

2.输入 `!6`, 执行 `history` 命令历史中第 6 行命令

```
[root@web ~]# !6
touch file.txt
```

3.使用 `!cat`, 调用 `history` 命令历史最近一次执行过的 `cat` 命令

```
[root@web ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
[root@web ~]# !cat
cat /etc/sysconfig/network-scripts/ifcfg-eth0
```

1.4.6.4 命令别名alias

命令别名将用户经常使用的复杂命令简单化, 可以用 `alias 别名名称='命令'` 创建属于自己的命令别名，若要取消一个命令别名，则是用 `unalias 别名名称`；

1.定义临时别名，`wk` 为查看 `eth0` 网卡别名

```
[root@web ~]# alias wk='ifconfig'
[root@web ~]# wk
```

2.如果定义命令本身，会执行什么?

```
[root@web ~]# alias ifconfig='ifconfig eth0'

#绝对路径执行，调用命令本身
[root@web ~]# /sbin/ifconfig

#通过\转义字符，调用命令本身
[root@Web ~]# \ifconfig
```

3.取消别名

```
[root@web ~]# unalias ifconfig
```

4.永久生效，`/etc/bashrc`

```
[root@web ~]# echo "alias ifconfig='ifconfig eth0'" >> /etc/bashrc
```

1.4.6.5 帮助手册help

1.命令 `--help` 帮助

```
[root@web ~]# ls --help
用法: ls [选项]...[文件]...
```

2.命令 `man` 手册

```
# man ls    #查看ls命令的手册
```

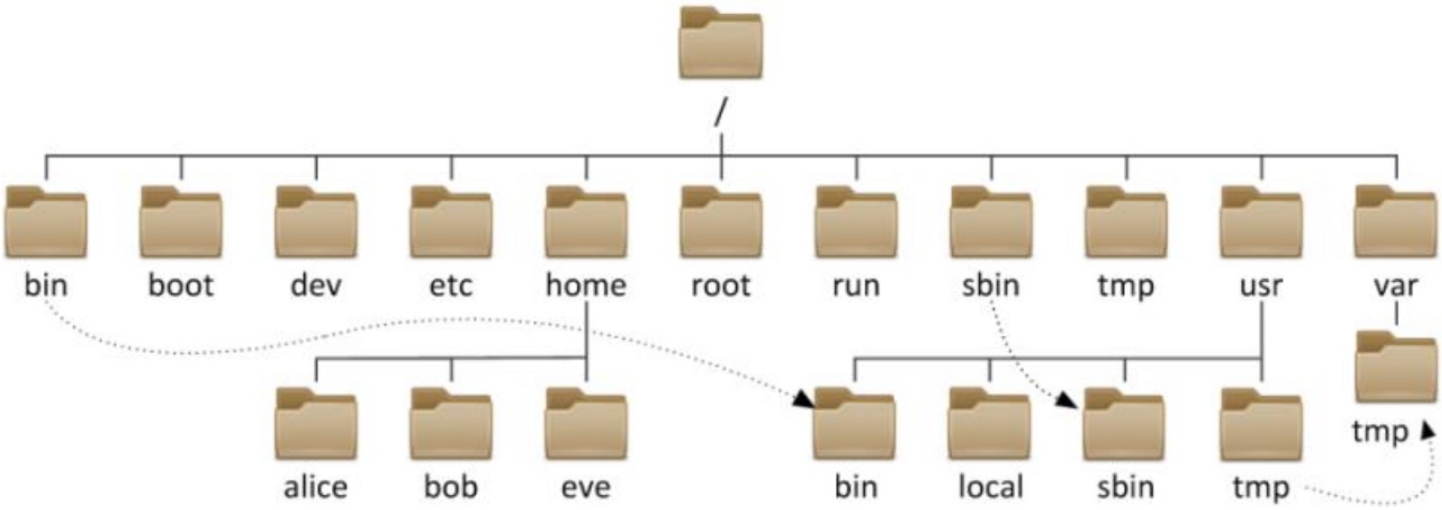
3. `linux` 命令大全 `url` 传送门
linux命令大全
linux命令手册

1.5 linux文件管理

谈到 `Linux` 文件管理，首先我们需要了解的就是，我们要对文件做些什么事情?
其实无非就是对一个文件进行、创建、复制、移动、查看、编辑、压缩、查找、删除、等等
如：当我们想修改系统的主机名称，是否应该知道文件在哪，才能去做对应的修改?

1.5.1 linux目录结构

几乎所有的计算机操作系统都是使用目录结构组织文件。具体来说就是在一个目录中存放子目录和文件, 而在子目录中又会进一步存放子目录和文件，以此类推形成一个树状的文件结构，由于其结构很像一棵树的分支, 所以该结构又被称为“目录树”;
`Windows`： 以多根的方式组织文件 `C:\ D:\`
`Linux`： 以单根的方式组织文件 `/`



1.5.1.1 命令相关目录/bin

- 存放命令相关的目录
  - `/bin` 普通用户使用的命令 `/bin/ls, /bin/date`
  - `/sbin` 管理员使用的命令 `/sbin/service`

1.5.1.2 用户家相关目录/home

- 存放用户相关数据的家目录，比如 `windows` 不同的用户登陆系统显示的桌面背景不一样
  - `/home` 普通用户的家目录, 默认为 `/home/username`
  - `/root` 超级管理员root的家目录, 普通用户无权操作

1.5.1.3 系统文件目录/usr

- 存放系统相关文件的目录
  - `/usr` 相当于 `C:\Windows`
  - `/usr/local` 软件安装的目录，相当于 `C:\Program`
  - `/usr/bin/` 普通用户使用的应用程序(重要)
  - `/usr/sbin` 管理员使用的应用程序(重要)
  - `/usr/lib` 库文件 `Glibc 32bit`
  - `/usr/lib64` 库文件 `Glibc 64bit`

1.5.1.4 系统启动目录/boot

- 存放系统启动时内核与grub引导菜单
  - /boot 存放的系统启动相关的文件，如： `kernel, grub` (引导装载程序)

1.5.1.5 配置文件目录/etc

- /etc存放系统配置文件目录，后续所有服务的配置都在这个目录中

    - `/etc/sysconfig/network-script/ifcfg-`，网络配置文件
    - `/etc/hostname` 系统主机名配置文件

1.5.1.6 设备相关目录/dev

- /dev存放设备文件的目录，比如硬盘，硬盘分区，光驱，等等
    - `/dev/null` 黑洞设备，只进不出。类似于垃圾回收站
    - `/dev/random` 生成随机数的设备
    - `/dev/zero` 能源源不断的产生数据，类似于取款机，随时随地取钱*

1.5.1.7 可变的目录/var

- /var，存放一些变化文件，比如/var/log/下的日志文件
- /var/tmp，进程产生的临时文件
- /tmp，系统临时目录(类似于公共厕所)

1.5.1.8 虚拟系统目录/proc

- 虚拟的文件系统 (如对应的进程停止则/proc下对应目录则会被删除)
    - /proc，反映当前系统正在运行进程的实时状态，类似于汽车在运行过程中的仪表板，能够看到汽车的油耗、时速、转向灯、故障等等

1.5.2 文件路径定位

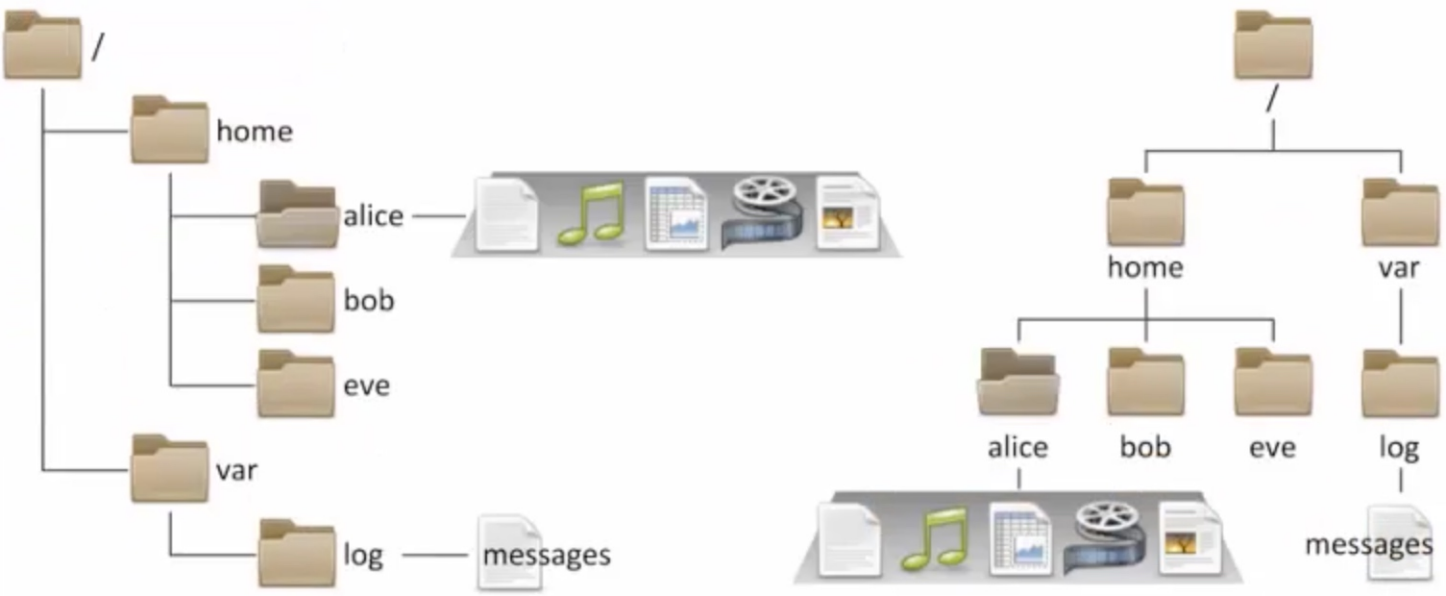
在我们开始操作文件之前，首先需要对文件的路径进行定位。

1.5.2.1 为什么要进行定位

- 你要在哪个目录下创建文件?
- 你要将文件复制到什么地方?
- 你要删除的文件在什么地方?

1.5.2.2 如何对文件进行定位

比如: `/etc/hostname` 整个文件中包含文件名称以及文件所在的位置，我们将这个叫做路径，也就是说我们是通过路径对文件进行定位。例：下图所示的 `message` 所在的路径是?



FQ1：`/home/boy/file` 和 `/home/girl/file` 是否是同一个文件?
FQ2：`/abc/test` 和 `abc/test` 是一样的吗?

1.5.2.3 绝对路径与相对路径

- 绝对路径: 只要从/开始的路径，比如 `/home/alice/file`
- 相对路径: 相对于当前目录来说，比如 `a.txt ./a.txt ../bob/a.mp3`
- `.` 和 `..` 的是什么意思
  - 一个点代表当的是当前目录；
  - 两个点代表的是当前目录的上层目录；
- 小结：所谓的(.)和(..)目录实际上属于相对路径的一种表示形式；

1.5.2.4 路径切换命令cd

例子如下：

```
# cd 绝对路径 cd /etc/hostname
# cd 相对路径 cd test/abc cd . cd ..
-------------------------------------
# cd      #切换目录，例: cd /etc
# cd -    #切换回上一次所在的目录
# cd ~    #切换回当前用户的家目录,注意:root和普通用户是否有所不同吗?
# cd .    #代表当前目录，一般在拷贝、移动等情况下使用 cp /etc/hostname ./
# cd ..   #切换回当前目录的上级目录
```

1.5.3 linux基础命令

1.5.3.1 文件操作类命令

1.5.3.1.1 touch文件创建

```
# touch file                      #无则创建,有则修改时间
# touch file2 file3
# touch /home/od/file4 file5
# touch file{a,b,c}               #{}集合, 等价 touch a b c
# touch file{1..10}
# touch file{a..z}
```

1.5.3.1.2 mkdir目录创建

```
# 选项: -v 显示详细信息  -p 递归创建目录
# mkdir dir1
# mkdir /home/ob/dir1 /home/ob/dir2
# mkdir -v /home/ob/{dir3,dir4}
# mkdir -pv /home/ob/dir5/dir6
# mkdir -pv /home/{ob/{diu,but},boy}
```

1.5.3.1.3 tree显示目录结构

```
# 选项: -L: 显示目录树的层级
# tree /home/ob/    #显示当前目录下的结构
/home/ob/
├── but
├── dir1
├── dir2
├── dir3
├── dir4
├── dir5
│   └── dir6
└── diu
```

1.5.3.1.4 cp文件或目录复制

```
#选项:  -v:详细显示命令执行的操作 -r: 递归处理目录与子目录 -p: 保留源文件或目录的属性

# cp file /tmp/file_copy
# cp name /tmp/name          #不修改名称
# cp file /tmp/              #不修改名称
# cp -p file /tmp/file_p     #-p保持原文件或目录的属性
# cp -r  /etc/ /tmp/         #复制目录需要使用-r参数, 递归复制
# cp -rv /etc/hosts /etc/hostname /tmp  #拷贝多个文件至一个目录
# cp -rv /etc/{hosts,hosts.bak}
# cp -rv /etc/hosts{,-org}
```

1.5.3.1.5 mv文件移动命令

```
# mv file file1              #原地移动算改名
# mv file1 /tmp/             #移动文件至tmp目录
# mv /tmp/file1 ./           #移动tmp目录的文件至当前目录
# mv dir/ /tmp/              #移动目录至/tmp目录下

# touch file{1..3}
# mv file1 file2 file3 /opt/    #移动多个文件或至同一个目录

# mkdir dir{1..3}
# mv dir1/ dir2/ dir3/ /opt     #移动多个目录至同一个目录
```

1.5.3.1.6 rm文件或目录删除

```
#选项：-r: 递归 -f: 强制删除 -v: 详细过程
# rm  file.txt      #删除文件，默认rm存在alias别名，rm -i所以会提醒是否删除文件
# rm -f file.txt    #删除文件，不提醒
---------------------------------
# rm -r dir/        #递归删除目录，会提示
# rm -rf dir/       #强制删除目录,不提醒(慎用)
---------------------------------
#1.rm删除示例
# mkdir /home/dir10
# touch /home/dir10/{file2,file3,.file4}
# rm -f /home/dir10/*  //不包括隐藏文件
# ls /home/dir10/ -a
. .. .file4
---------------------------------
#2.rm删除示例2
# touch file{1..10}
# touch {1..10}.pdf
# rm -rf file*
# rm -rf *.pdf
```

1.5.3.2 文件查看类命令

1.5.3.2.1 cat命令

```
#------cat
# cp /etc/passwd ./pass
# cat pass      #正常查看文件方式
# cat -n pass   #-n显示文件有多少行
```

```
# cat -A pass   #查看文件的特殊符号,比如文件中存在tab键
# tac pass      #倒序查看文件
```

1.5.3.2.2 less-more命令

```
#------less、more
# less /etc/services    #使用光标上下翻动，空格进行翻页，q退出
# more /etc/services    #使用回车上下翻动，空格进行翻页，q退出
```

1.5.3.2.3 head-tail 命令

```
#------head
# head pass     #查看头部内容，默认前十行
# head -n5 pass #查看头部5行，使用-n指定

#------tail
# tail pass
# tail -20 /var/log/secure
# tail -f /var/log/messages #-f查看文件尾部的变化
# tailf /var/log/messages   #查看文件尾部的变化
```

1.5.3.2.4 grep过滤数据

```
#------grep过滤文件内容
# grep "^root" pass     #匹配以root开头的行
# grep "bash$" pass     #匹配以bash结尾的行
# grep -v "ftp" pass    #匹配除了包含ftp的内容，其他全部打印
# grep -i "ftp" pass    #忽略大小写匹配
# grep  -Ei "sync$|ftp" pass    #匹配文件中包含sync结尾或ftp字符串
# grep -n -A 2 "Failed" /var/log/secure #匹配/var/log/secure文件中Failed字符串,并打印它的下2行
# grep -n -B 2 "Failed" /var/log/secure #匹配/var/log/secure文件中Failed字符串,并打印它的上2行
# grep -n -C 2 "Failed" /var/log/secure #匹配/var/log/secure文件中Failed字符串,并打印它的上下2行
```

1.5.3.3 文件下载类命令

1.5.3.3.1 wget命令

```
#CentOS7 系统最小化安装默认没有wget命令，需要进行安装
# yum install wget -y
```

```
#下载互联网上的文件至本地
#wget http://mirrors.aliyun.com/repo/Centos-7.repo

#将阿里云的centos-7.repo下载到/etc/yum.repos.d/并改名为CentOS-Base.repo  -O参数指定
# wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7
.repo
```

1.5.3.3.2 curl命令

```
#仅查看这个url地址的文件的内容
# curl  http://mirrors.aliyun.com/repo/Centos-7.repo

#将阿里云的centos-7.repo下载到/etc/yum.repos.d/并改名为CentOS-Base.repo  -o参数指定
# curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7
.repo


#练习：请下载一个图片至于/opt目录下(不要修改名称),最少使用2中方式，URl地址为：http://fj.x
uliangwei.com/public/ks.jpeg
#1.wget
[root@www ~]# cd /opt
[root@www opt]# wget http://fj.xuliangwei.com/public/ks.jpeg
[root@www ~]# wget -O /opt/ks.jpeg  http://fj.xuliangwei.com/public/ks.jpeg

#2.curl
[root@www ~]# curl -o /opt/ks2.jpeg http://fj.xuliangwei.com/public/ks.jpeg
```

1.5.3.3.3 rz-sz命令

```
# yum install lrzsz -y  #不安装软件则无法执行该命令

# rz            #只能上传文件，不支持上传文件夹，不支持大于4个G上传，也不支持断电续传
# sz /path/file #只能下载文件，不支持下载文件夹
```

1.5.3.4 字符处理类命令

1.5.3.4.1 sort命令

在有些情况下，需要对应一个无序的文本文件进行数据的排序，这时就需要使用sort进行排序了。

```
sort [OPTION]... [FILE]...
# -r：倒序 -n：按数字排序 -t：指定分隔符(默认空格) -k：指定第几列，指定几列几字符（指定1,1
  3.1,3.3）
```

```
#1.首先创建一个文件，写入一写无序的内容
[root@www ~]# cat >> file.txt <<EOF
b:3
c:2
a:4
e:5
d:1
f:11
EOF

#2.使用sort下面对输出的内容进行排序
[root@www ~]# sort file.txt
a:4
b:3
c:2
d:1
e:5
f:11

#结果并不是按照数字排序，而是按字母排序。
#可以使用-t指定分隔符，使用-k指定需要排序的列。
[root@www ~]# sort -t ":" -k2 sort.txt
d:1
f:11 #第二行为什么是11？不应该按照顺序排列？
c:2
b:3
a:4
e:5

#按照排序的方式，只会看到第一个字符,11的第一个字符是1，按照字符来排序确实比2小。
#如果想要按照数字的方式进行排序，需要使用 -n参数。
[root@www ~]# sort -t ":" -n -k2 p.txt
d:1
c:2
b:3
a:4
e:5
f:11


#测试案例，下载文件http://fj.xuliangwei.com/public/ip.txt，对该文件进行排序
[root@www ~]# sort -t. -k3.1,3.1nr -k4.1,4.3nr ip.txt
```

1.5.3.4.2 uniq命令

如果文件中有多行完全相同的内容，当前是希望能删除重复的行，同时还可以统计出完全相同的

行出现的总次数, 那么就可以使用 `uniq` 命令解决这个问题(但是必须配合 `sort` 使用)。

```
uniq [OPTION]... [INPUT [OUTPUT]]
#选项: -c  计算重复的行

#1.创建一个file.txt文件:
[root@www ~]# cat file.txt
abc
123
abc
123
#2.uniq需要和sort一起使用, 先使用sort排序, 让重复内容连续在一起
[root@www ~]# sort file.txt
123
123
abc
abc
#3.使用uniq去除相邻重复的行
[root@www ~]# sort file.txt |uniq
123
abc
#4.-c参数能统计出文件中每行内容重复的次数
[root@www ~]# sort file.txt |uniq -c
      2 123
      2 abc

#面试题: 请统计分析如下日志, 打印出访问最高前10的IP
```

2.5.4 wc命令

```
wc [OPTION]... [FILE]...
#选项: -l显示文件行数 -c显示文件字节 -w显示文件单词

# wc -l /etc/fstab      #统计/etc/fstab文件有多少行
# wc -l /etc/services   #统计/etc/services 文件行号

#练习题: 过滤出/etc/passwd以nologin结尾的内容, 并统计有多少行


# 扩展统计文件行号的方法
# grep -n ".*" /etc/services  | tail -1
# cat -n /etc/services  | tail -1
# awk '{print NR $0}' /etc/services | tail -1
```

1.5.3.5 练习讲解

1.5.3.5.1 练习1

- 分析如下日志，统计每个域名被访问的次数。

```
[root@student tmp]# cat web.log
http://www.example.com/index.html
http://www.example.com/1.html
http://post.example.com/index.html
http://mp3.example.com/index.html
http://www.example.com/3.html
http://post.example.com/2.html

# awk -F '/' '{print $3}' web.log|sort -rn|uniq -c
# cut -d / -f3 web.log|sort -rn|uniq -c
```

1.5.3.5.2 练习2

- 使用awk取出系统的IP地址，使用多种方法实现
  - 1.我要取的值在哪里
  - 2.如何缩小取值范围(行)
  - 3.如何精确具体内容(列)

1.5.3.5.3 练习3

- 将 `/etc/sysconfig/selinux` 文件中的 `SELINUX=enforcing` 替换成 `SELINUX=disabled`

1.5.3.5.4 练习4

- 将 `/etc/passwd` 文件中的第一行中的第一列和最后一列位置进行交换。

1.5.3.5.5 练习5

- 现有 `1-100` 个文件，需要保留 `75，76，78` 三个文件，其余全部删除。 `grep、awk、sed`

1.5.4 linux文件属性

1.5.4.1 文件属性

当我们使用 `ls -l` 列目录下所有文件时，通常会以长格式的方式显示，其实长格式显示就是我们 `Windows` 下看到的文件详细信息，我们将其称为文件属性，那整个文件的属性分为十列。

```
[root@www ~]# ls -l ks.cfg
-rw-------. 1 root root 4434 May 30 13:58 ks.cfg
```

```
-rw-------. ①:第一个字符是文件类型，其他则是权限
1           ②:硬链接次数
root        ③:文件属于哪个用户
root        ④:文件属于哪个组
4434        ⑤:文件大小
May30 13:58 ⑥⑦⑧:最新修改的时间与日期
ks.cfg      ⑨:文件或目录名称
```

1.5.4.2 文件类型

通常我们使用颜色或者后缀名称来区分文件类型，但很多时候不是很准确，所以我们可以通过 `ls -l` 以长格式显示一个文件的属性，通过第一列的第一个字符来近一步的判断文件具体的类型。

```
[root@www ~]# ll -d /etc/hosts /tmp /bin/ls  /dev/sda /dev/tty1 /etc/grub2.cfg /dev
/log /run/dmeventd-client
-rwxr-xr-x.  1 root root 117656 Jun 30  2016 /bin/ls
srw-rw-rw-.  1 root root      0 Jan 20 10:35 /dev/log
brw-rw----.  1 root disk   8, 0 Jan 20 10:36 /dev/sda
crw--w----.  1 root tty    4, 1 Jan 20 10:36 /dev/tty1
lrwxrwxrwx.  1 root root     22 Jan 13 11:31 /etc/grub2.cfg -> ../boot/grub2/grub.c
fg
-rw-r--r--.  1 root root    199 Jan 20 11:03 /etc/hosts
drwxrwxrwt. 61 root root   8192 Jan 21 13:01 /tmp
```

- 文件类型说明

文件类型字母	类型含义
-	普通文件(文本, 二进制, 压缩, 图片, 日志等)
d	目录文件
b	设备文件(块设备)存储设备硬盘 /dev/sda, /dev/sr0
c	设备文件(字符设备)，终端 /dev/tty1
s	套接字文件，进程与进程间的一种通信方式(socket)
l	链接文件

- 但有些情况下，我们无法通过ls -l文件的类型，比如: 一个文件，它可能是普通文件、也可能是压缩文件、或者是命令文件等，那么此时就需要使用file来更加精准的判断这个文件的类型

```
[root@www ~]# file /etc/hosts
/etc/hosts: ASCII text

[root@www ~]# file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (u
ses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=aa7ff68f13de25936a098016243ce
57c3c982e06, stripped

[root@www ~]# file /dev/sda
/dev/sda: block special

[root@www ~]# file /dev/tty1
/dev/tty1: character special

[root@www ~]# file /etc/grub2.cfg
/etc/grub2.cfg: broken symbolic link to `../boot/grub2/grub.cfg

[root@www ~]# file /home
/home: directory
```

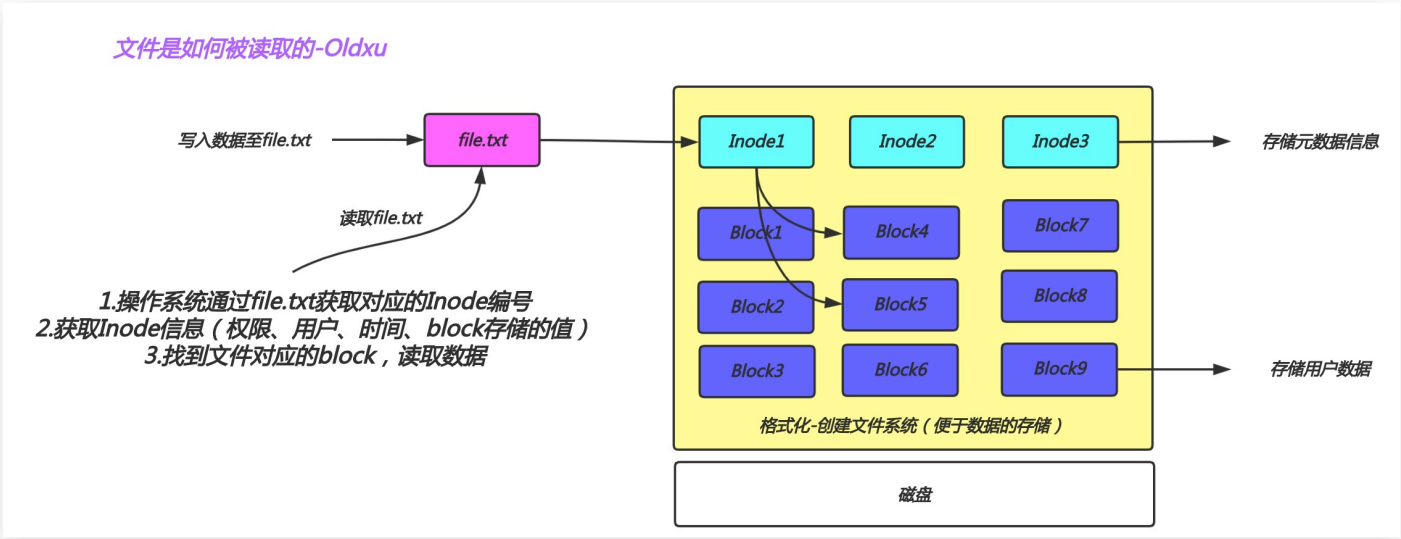
PS: Linux文件扩展名不代表任何含义，仅为了我们能更好的识别该文件是什么类型。

1.5.5 linux链接文件

1.5.5.1 Inode与Block

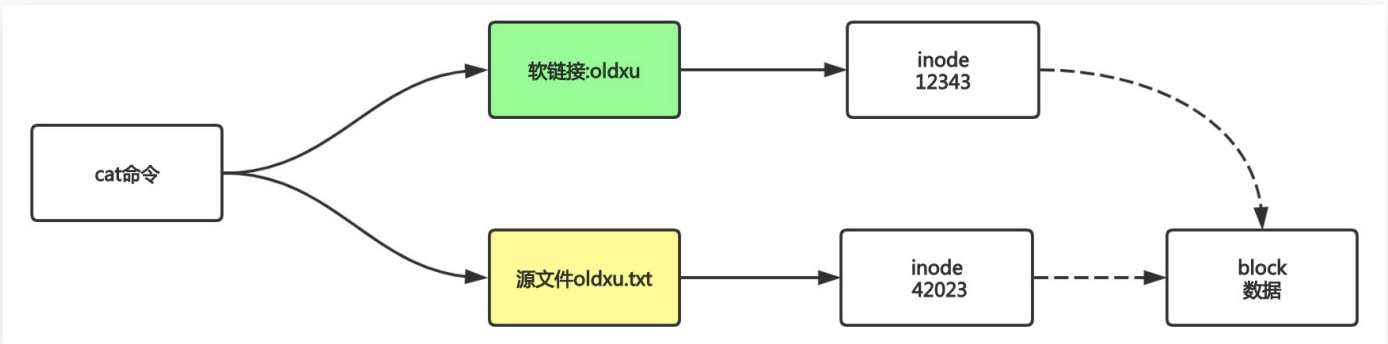
- 文件有文件名与数据，在Linux上被分成两个部分：数据`data`与文件元数据`metadata`
    - 1.数据`data block`，数据块是用来记录文件真实内容的地方，我们将其称为`Block`
    - 2.元数据`metadata`，用来记录文件大小、创建时间、所有者等信息，我们将其称为`Inode`
    - 3.需要注意：Inode并不包含文件名称，inode仅包含文件的元数据信息，具体来说有以下内容：
        - 文件的字节数
        - 文件的User ID Group ID
        - 文件的读、写、执行权限
        - 文件的时间戳
        - 链接数，即有多少文件名指向这个inode
        - 文件数据block的位置
- 每个`inode`都是一个编号，操作系统是通过Inode来识别不同的文件。
    - 对于系统来说，文件名只是inode便于识别的别名，或者绰号。（便于我们人识别。）

- 表面上，用户是通过文件名打开的文件，实际上系统内部这个过程分为如下三步：
  - 首先，系统找到这个文件名对应的inode编号
  - 其次，通过inode编号，获取inode信息
  - 最后，根据inode信息，找到文件数据所在的block，读出数据。

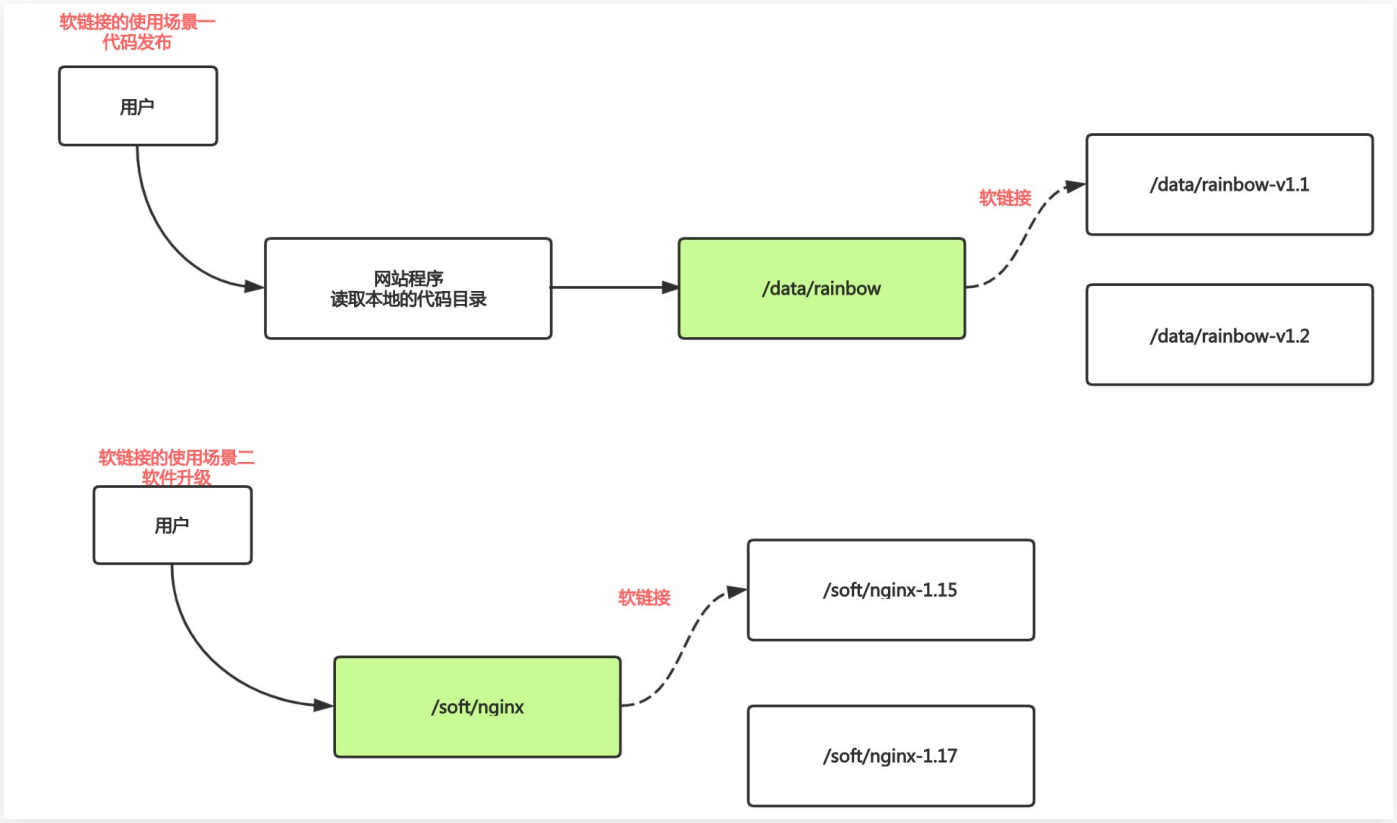


1.5.5.2 软连接

- 什么是软连接：
  - 软链接相当于 `Windows` 的快捷方式，软链接文件会将 `inode` 指向源文件的 `block` 当我们访问这个软链接文件时，其实访问的是源文件本身；



- 软连接使用场景
  - 软件升级
  - 代码发布



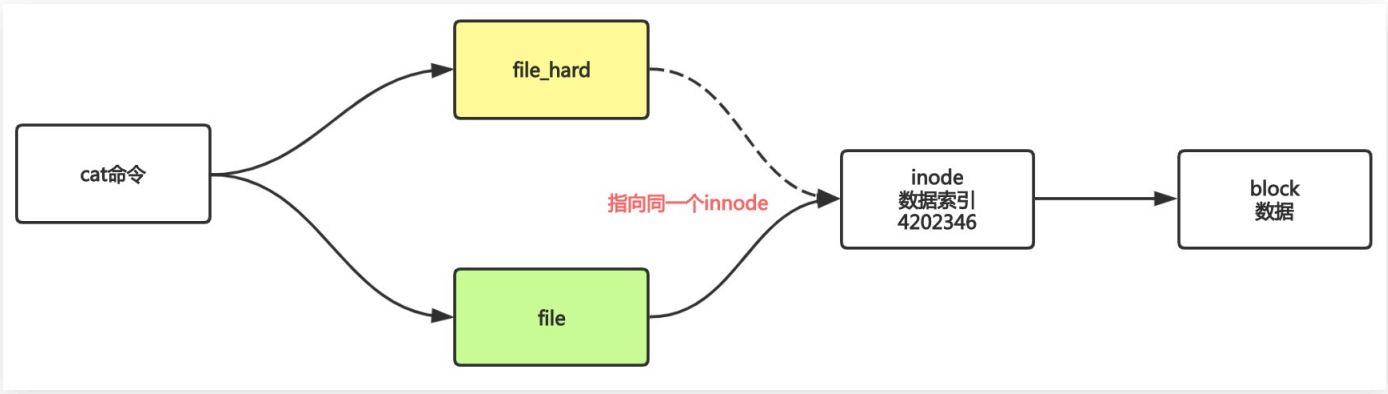
- 软链接场景实践

```
#1.准备网站1.1版本代码
[root@www ~]# mkdir /data/rainbow-v1.1 -p
[root@www ~]# echo "123" > /data/rainbow-v1.1/index.html

#2.创建软链接
[root@www ~]# ln -s /data/rainbow-v1.1/ /data/rainbow
[root@www ~]# ll /data/
drwxr-xr-x. 2 root root  6 3月   5 12:23 dir
lrwxrwxrwx. 1 root root 19 3月  10 12:09 rainbow -> /data/rainbow-v1.1/
drwxr-xr-x. 2 root root 24 3月  10 12:09 rainbow-v1.1

#3.检查网站程序
[root@www ~]# cat /data/rainbow/index.html
123

#4.新更新一个网站的程序代码
[root@www ~]# mkdir /data/rainbow-v1.2
[root@www ~]# echo "456" > /data/rainbow-v1.2/index.html

#5.升级
[root@www ~]# rm -f /data/rainbow && ln -s /data/rainbow-v1.2/ /data/rainbow
[root@www ~]# cat /data/rainbow/index.html
456
```

```
#6.回退
[root@www ~]# rm -f /data/rainbow && ln -s /data/rainbow-v1.1/ /data/rainbow
[root@www ~]# cat /data/rainbow/index.html
123
```

1.5.5.3 硬连接

硬链接类似于超时有多个门，无论从哪个门进入，看到的内容都是一样的。如果关掉一扇门，那影响进入超市？回到系统中，我们对硬链接的解释: 不同的文件名指向同一个 `inode` ，简单的说就是指向同一个真实的数据源。



- 硬链接与软链接区别
  - 1）ln命令创建硬链接，ln -s命令创建软链接
  - 2）目录不能创建硬链接，并且硬链接不可以跨越分区系统；
  - 3）软链接支持对目录创建，同时也支持跨越分区系统；
  - 4）硬链接文件与源文件的inode相同，软链接文件与源文件inode不同；
  - 5）删除软链接文件，对源文件及硬链接文件无任何影响；
  - 6）删除文件的硬链接文件，对源文件及链接文件无任何影响；
  - 7）删除链接文件的源文件，对硬链接无影响，会导致软链接失效；
  - 8）删除源文件及其硬链接文件，整个文件会被真正的删除；

1.6 linux编辑工具vim

1.6.1 vim基础

1.6.1.1 什么是vim

`vi` 和 `vim` 是 `Linux` 下的一个文本编辑工具。(可以理解为 `windows` 的记事本，或 `Notepad++`

1.6.1.2 为什么需要vim

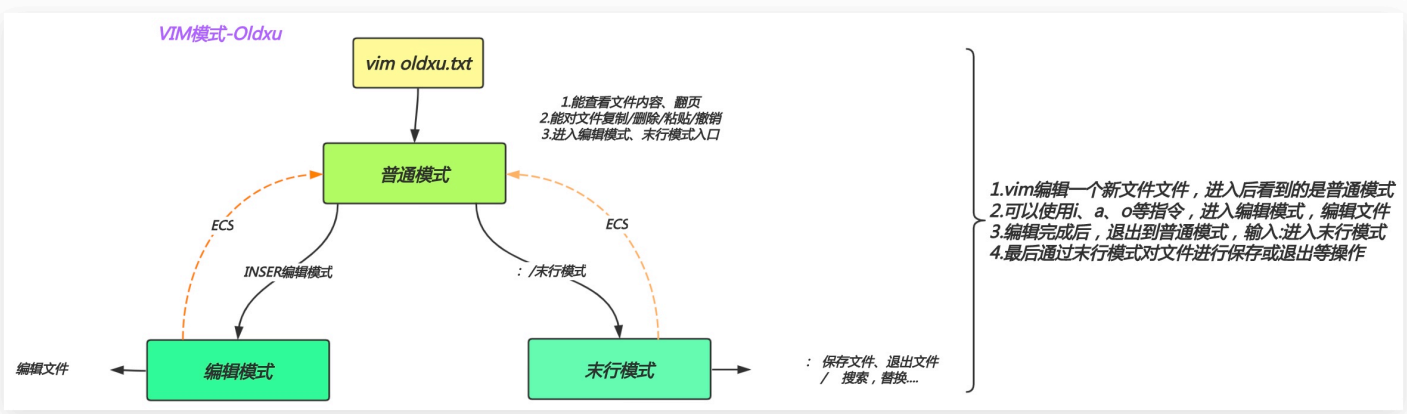
因为 `Linux` 一切皆为文件，而我们工作最多的就是修改某个服务的配置 (其实就是修改文件内容)。
也就是说如果没有 `vi/vim` 我们很多工作都无法完成。所以 `vim` 是学习 `linux` 最重要的命令之一

1.6.1.3 vi与vim的区别

`vi` 和 `vim` 都是文本编辑器，只不过 `vim` 是 `vi` 的增强版，比 `vi` 多了语法高亮显示，其他编辑功能几乎无差，所以使用 `vi` 还是 `vim` 取决个人习惯。
由于前期我们采用最小化安装操作系统所以没有 `vim` 命令，可以使用 `yum install vim` 进行安装

1.6.1.4 如何使用vim

在使用 `VIM` 之前，我们需要先介绍下 `VIM` 的三种模式： 普通模式、编辑模式、末行模式
每种模式分别支持多种不同的快捷键，要想高效率地操作文本，就必须先搞清这三种模式的操作区别以及模式之间的切换方法。



- VIM模式三种模式介绍
  - 1.普通模式：主要是控制光标移动，可对文本进行复制、粘贴、删除等工作。
    - 使用 `vim filename` 编辑一个文件时，一进入该文件就是普通模式了。
    - 在这个模式下，可以进行光标移动、复制、删除、粘贴操作。
  - 2.编辑模式：主要进行文本内容编辑和修改
    - 从普通模式进入编辑模式，只需你按一个键即可 `i, I, a, A, o, O`
    - 当进入编辑模式时，会在屏幕的最下一行会出现 `INSERT` 标记
    - 从编辑模式回到普通模式只需要按键盘左上方的 `ESC` 键即可。
  - 3.末行模式：主要用于保存或退出文本。
    - 在普通模式下，输入 ":" 或者 "/" 即可进入末行模式。
    - 在命令该模式下，可进行的操作有，显示行号、搜索、替换、保存、退出。
  - 4.小结：vim编辑打开文件整体流程如下:
    - 1.默认打开文件处于普通模式

        - 2.从普通模式切换至编辑模式需要使用a、i、o
        - 3.编辑模式修改完毕后需要先使用ECS返回普通模式
        - 4.在普通模式输入":"或"/"进入末行模式，可实现文件的保存与退出。
    - 注意：在vim中，无法直接从编辑模式切换到末行模式。

1.6.2 VIM模式使用

1.6.2.1 普通模式

- 普通模式，主要用于光标移动，复制，粘贴，删除，替换；

```
#1.命令光标跳转
G        #光标跳转至末端
gg       #光标跳转至顶端
Ngg      #光标跳转至当前文件内的N行
$        #光标跳转至当前光标所在行的尾部
^|0      #光标跳转至当前光标所在行的首部
-----------------------------------------
#2.文件内容较多
ctrl+f  #往下翻页(行比较多)
ctrl+b  #往上翻页
-----------------------------------------
#3.复制与粘贴
yy       #复制当前光标所在的行
5yy      #复制当前光标以及光标向下4行

p(小写)   #粘贴至当前光标下一行
P(大写)   #粘贴至当前光标上一行
-----------------------------------------
#4.删除、剪贴、撤销
dd       #删除当前光标所在的行
4dd      #删除当前光标所在的行以及往下的3行
dG       #删除当前光标以后的所有行
D        #删除当前光标及光标以后的内容
x        #删除当前光标标记往后的字符
X        #删除当前光标标记往前的字符
dd & p  #剪贴、先删除dd(number dd)，后粘贴p
u        #撤销上一次的操作
-----------------------------------------
#5.替换
r        #替换当前光标标记的单个字符
R        #进入REPLACE模式，连续替换，ESC结束
```

1.6.2.2 编辑模式

- 编辑模式，主要用于编辑文件；

```
i   #进入编辑模式，光标不做任何操作
a   #进入编辑模式，将当前光标往后一位
o   #进入编辑模式，并在当前光标下添加一行空白内容
-----------------------------------------
I   #进入编辑模式，并且光标会跳转至本行的头部
A   #进入编辑模式，将光标移动至本行的尾部
O   #进入编辑模式，并在当前光标上添加一行空白内容
```

1.6.2.3 末行模式

- 末行模式，主要用于搜索, 保存, 退出文件；

```
#1.文件保存与退出
:w       保存当前状态
:w!      强制保存当前状态
:q       退出当前文档(文档必须保存才能退出)
:q!      强制退出文档不会修改当前内容
:wq      先保存，在退出
:wq!     强制保存并退出
:x       先保存，在退出
ZZ       保存退出，shfit+zz
:number 跳转至对应的行号
-----------------------------------------
#2.文件内容查找
/string #需要搜索的内容（查找）
n        #按搜索到的内容依次往下进行查找
N        #按搜索到的内容依次往上进行查找
-----------------------------------------
#3.文件内容替换
:1,5s#sbin#test#g    #替换1-5行中包含sbin的内容为test
:%s#sbin#test#g      #替换整个文本文件中包含sbin的替换为test
:%s#sbin#test#gc     #替换内容时时提示是否需要替换
-----------------------------------------
#4.文件内容另存
:w /root/test.txt  #将所有内容另存为/root/test.txt文件中
-----------------------------------------
#5.文件内容读入
:r  /etc/hosts  #读入/etc/hosts文件至当前光标下面
:5r /etc/hosts  #指定插入/etc/hosts文件至当前文件的第五行下面
```

1.6.2.4 视图模式

- 视图模式(从普通模式进入视图模式)，主要进行批量操作

```
ctrl+v  进入可视块模式，选中需要注释的行
    1.插入:按shift+i进入编辑模式,输入#,结束按ESC键
    2.删除:选中内容后，按x或者d键删除
    3.替换:选中需要替换的内容，按下r键,然后输入替换后的内容
----------------------------------------
shift+v 进入可视行模式，选中整行内容
    1.复制:选中行内容后按y键及可复制。
    2.删除:选中行内容后按d键删除。
```

1.6.3 VIM扩展知识

1.6.3.1 vim环境变量

1.环境变量临时生效

```
:set nu              #显示行号
:set ic              #忽略大小写，在搜索的时候有用
:set ai              #自动缩进
:set list            #显示制表符(空行、tab键)
:set no[nu|ic|ai…]  #取消临时设定的变量
```

2.环境变量永久生效。~/.vimrc 个人环境变量(优先级高) /etc/vimrc 全局环境变量

```
# vim  ~/.vimrc #当下次再打开文件自动显示行号并忽略大小写
set nu
set ic

#如果个人vim环境没有配置，则使用全局vim环境变量配置。
#如果个人vim环境和全局环境变量产生冲突，优先使用个人vim环境变量。
```

1.6.3.2 vimdiff文件比对

4.相同文件之间差异对比，通常用于对比修改前后差异

```
# diff      #文件对比
# vimdiff   #以vim方式打开两个文件对比，高亮显示不同的内容
```

1.6.3.3 vim异常退出处理

5.如果VIM非正常退出 （ctrl+z）挂起或强制退出终端没关闭VIM后

```
#假设打开filename文件被以外关闭，需要删除同文件名的.swp文件即可解决
# rm -f .filename.swp
```

1.6.4 VIM练习示例

1.6.4.1 vim练习示例1

- 需求：
  - 1.将/etc/passwd 复制到/root/目录下，并重命名为test.txt
  - 2.用vim打开test.txt并显示行号
  - 3.分别向下、向右、向左、向右移动5个字符，分别向下、向上翻两页
  - 4.把光标移动到行末，再移动到行首，移动到test.txt文件的最后一行，移动到文件的首行
  - 5.搜索文件中出现的 root 并数一下一共出现多少个,不区分大小写搜索
  - 6.把从第一行到第三行出现的root 替换成--dd--，然后还原上一步操作\
  - 7.把整个文件中所有的root替换成--dd--
  - 8.把光标移动到20行，删除本行，还原上一步操作
  - 9.删除第19行，还原上一步操作
  - 10.删除从5行到10行的所有内容，还原上一步操作
  - 11.复制2行并粘贴到11行下面，还原上一步操作（按两次u）
  - 12.复制从11行到15行的内容并粘贴到8行上面，还原上一步操作（按两次u）
  - 13.把13行到18行的内容移动文件的尾部，还原上一步操作（按两次u）
  - 14.光标移动到首行，把/sbin/nologin改成/bin/bash
  - 15.在第一行下面插入新的一行，并输入"# Hello!"
  - 16.保存文档并退出

1.6.4.2 vim练习示例2

- 需求：
  - 1.使用vim打开proxy.conf文件
  - 2.修改Listen为listen小写，并将8080修改为80
  - 3.修改ServerName为server_name小写。
  - 4.修改vim.EXAMPLE.com为vim.example.com
  - 5.在server_name行下插入一行 root /code;
  - 5.复制5-14行的内容，然后将其粘贴到14行下面
  - 6.删除与proxy_set_header相关的两行全部删除
  - 7.如上操作完成后，在13-20行前面加上#号

- 8.删除21-23的行，然后保存当前文件

```
[root@www ~]# cat proxy.conf
server {
    Listen 8080;
    Server_Name vim.EXAMPLE.com;
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forward-for;
        proxy_intercept_errors on;
        proxy_next_upstream error timeout;
        proxy_next_upstream_timeout 3s;
        proxy_next_upstream_tries 2;
        error_page 500 502 403 404 = /proxy_error.html;
    }
    location = /proxy_error.html {
        root /code/proxy;
    }
}
```