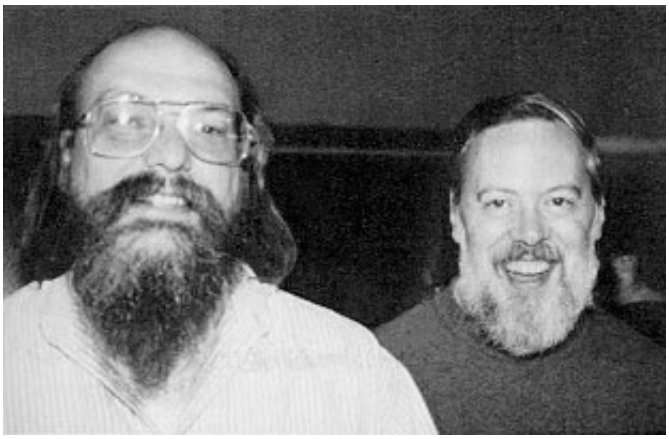
Linux

Auth : 张旭
Date : 2018-03-15
Email: zhangxu@1000phone.com

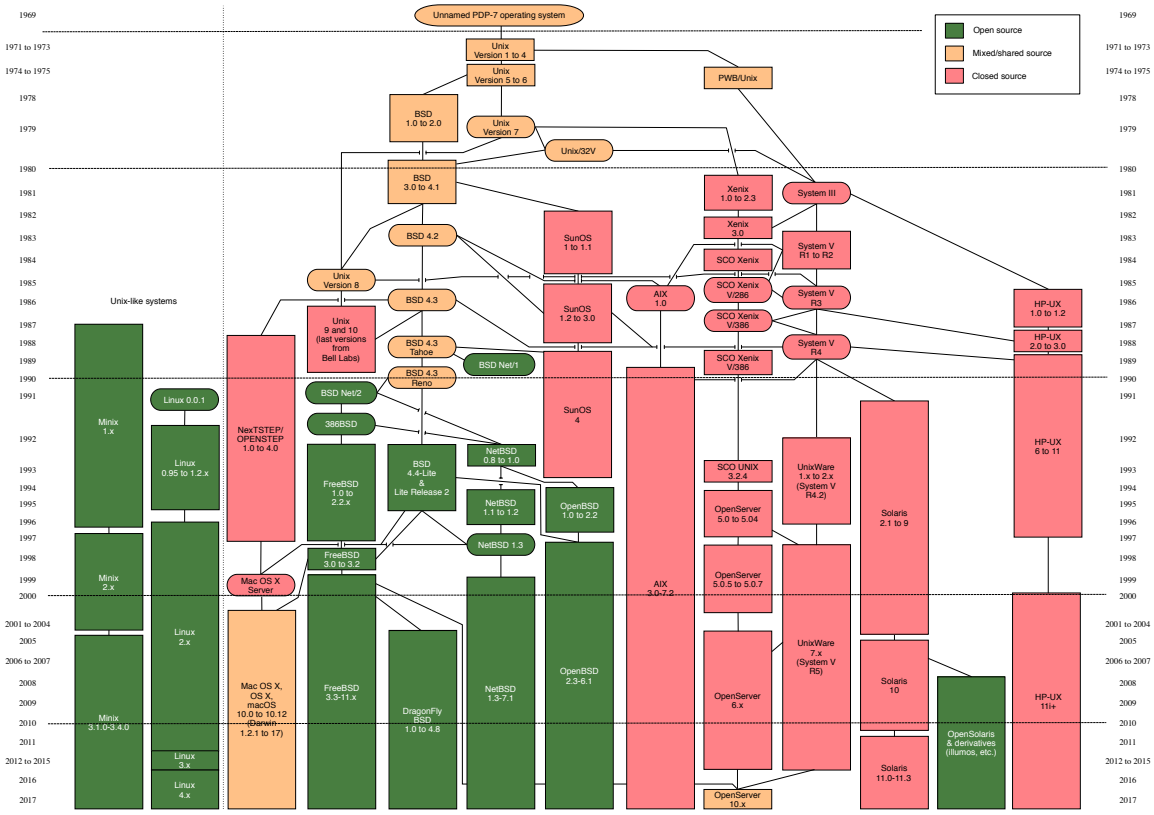
Linux发展史及流行版本简介

1. **Unix**: 一场关于 “太空旅行游戏” 的游戏
    - 
    - Ken Thompson、Dennis Ritchie
    - 贝尔实验室
    - Ken Thompson: Unix、C 语言、Go 语言、正则表达式、UTF-8, 全都跟这个老爷子有关
    - Unix 哲学
        - 小即是美
        - 程序应该只关注一个目标, 并尽可能把它做好
        - 让程序能够互相协同工作, 通过小程序协作完成大的功能
        - 避免使用可定制性低下的用户界面
2. **GNU** is Not Unix!
    -

    - Richard Stallman
    - 开源运动
        - GPL 协议: 使用我的代码, 你必须也得开源
        - OpenSource : 开源不代表我没有版权
        - FreeSoftware : 自由软件不是免费软件
3. **Linux**
    - 

- 有个性的芬兰大学生: Linus Torvalds
- 吉祥物 Tux
- Linux 标准发音
- 从 KDE 到 GNOME

4. **重要的发行版**
   - RedHat: 最成功的商用 Linux
   - CentOS: 社区版的 RedHat
   - Fedora: 个人版的 RedHat
   - Debian: 纯粹的自由软件构件的发行版, 拥有最大的开源软件库
   - Ubuntu: 友好的桌面版 Linux
   - Gentoo: 一切从源码开始手动安装, 性能超高, 非常稳定
   - Arch: 省去编译, 手动安装一切, 性能同样优异
   - Deepin: 国人制作的发行版, QQ、WPS、搜狗输入法, 除了游戏外能满足你的全部习惯
5. **Unix Family**



Linux 的安装

- 虚拟机下安装 Ubuntu
- 网络、磁盘、共享文件夹、vm tools
- 一键安装 ubuntu 下的常用库

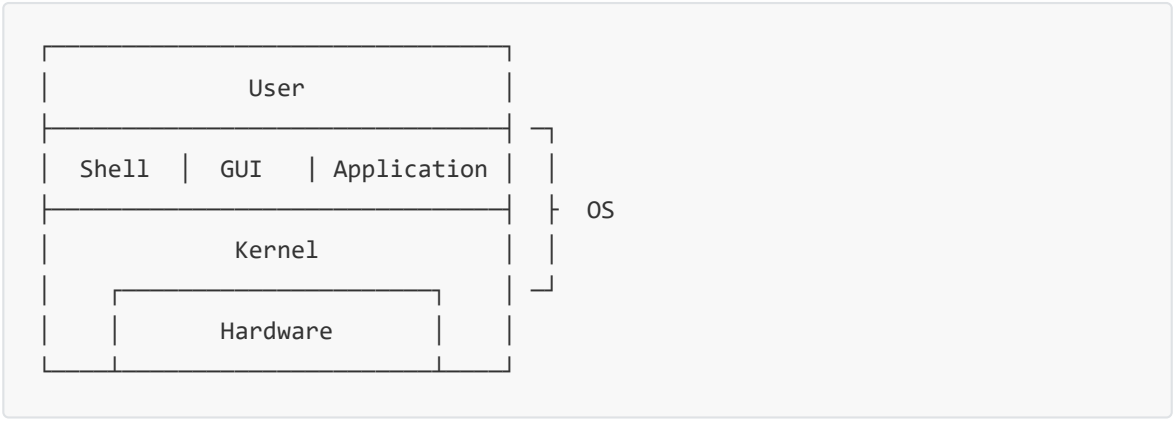
```
$ apt-get install -y man gcc make sudo lsof ssh openssl tree vim dnsutils
iputils-ping net-tools psmisc sysstat curl telnet traceroute wget libbz2-dev
libpcre3 libpcre3-dev libreadline-dev libsqlite3-dev libssl-dev zlib1g-dev git
mysql-server mysql-client zip p7zip
```

目录结构

```
/               系统根目录
├── boot        启动目录
│
├── bin         系统可执行目录
├── sbin        系统管理员的可执行目录
├── usr         资源目录 (User System Resources)
│   ├── bin
│   └── sbin
│   └── local
│       ├── bin
│       └── sbin
├── opt         第三方开发的程序 (option), 意为"选装"
│
├── dev         设备目录
│   ├── null    无底洞 (丢弃一切写入其中的数据)
│   ├── zero    无限 0 数据流 (常用来产生一个特定大小的空白文件, 或安全销毁文件)
│   ├── shm     内存文件夹
│   ├── random  随机数发生器
│   └── urandom 非阻塞的随机数发生器
├── etc         配置 (et cetera)
├── proc        当前的进程、运行状态信息的目录
│
├── root        管理员家目录
├── home        用户的家目录
│
├── media       系统挂载设备
├── mnt         用户挂载设备 (mount)
│
├── lib         系统库
│
├── run         运行中的程序的日志文件
├── tmp         临时文件夹
└── var         可变的、临时性的文件, 各种系统日志文件
```

Shell 是什么鬼

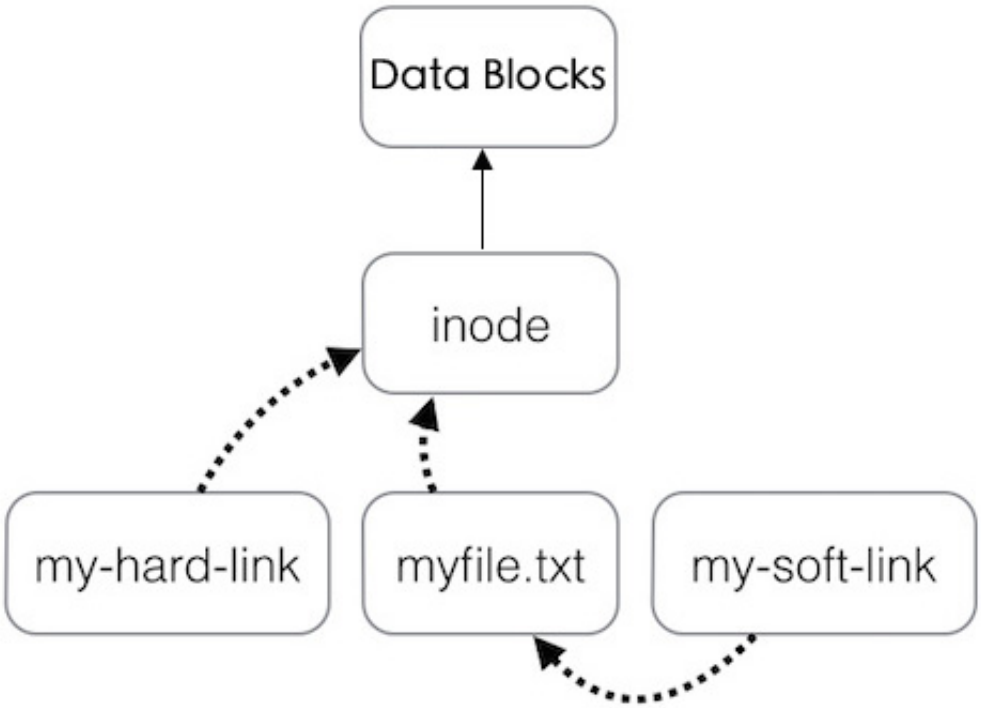
- Kernel



- CLI
  - sh
  - bash
  - zsh
- GUI
  - GNOME
  - KDE

日常操作

- `cd`, `ls`, `cp`, `mv`, `rm`, `mkdir`
  - cp / mv / rm : -i 询问 -f 强制 -n 不覆盖
  - cp / rm: -r 递归子文件夹
  - `mkdir -p a/b/c` 按层级创建
  - `mkdir -p a/{b,c}/{d,e,f}` 同一层级创建多个
- `pwd` 显示当前完整路径
- `ln -s [src_file] [dst_file]`



- `touch` 没有则创建，有则忽略
- `alias` 自定义别名
- `history`
  - bashrc 配置显示时间: `export HISTTIMEFORMAT="[%y-%m-%d_%T]  "`
  - 修改 bashrc 后使其生效: `source ~/.bashrc` 或 `. .bashrc`

Bash 快捷键

- `ctl + f` 前进一个字符
- `ctl + b` 后退一个字符
- `ctl + a` 回到行首
- `ctl + e` 回到行尾
- `ctl + w` 向左删除一个单词
- `ctl + u` 向左删除全部
- `ctl + k` 向右删除全部
- `ctl + y` 粘贴上次删除的内容
- `ctl + l` 清屏

进程管理

- ps : process status
- `ps -aux` 或 `ps ex` 查看进程
- `top -p PID1,PID2,PID3,...,PID20` 动态监控进程
- `free -m` 以 Mb 为单位查看内存
- `kill`

    - 给进程发送信号, 信号详情: `man signal`
    - `-1 (HUP)` 不间断重启
    - `-9 (KILL)` 强制杀死进程
    - `-15 (TERM)` 正常终止进程 (default)
- `pkill [ProcessName]` 按名字处理进程
- `killall [MatchedProcessName]` 处理名字匹配的进程
- `uptime` 查看系统状态

权限管理

- user 和 group
    - `groupadd admin`
    - `useradd -G adm,sudo -m -s /bin/bash bigcat`
- `sudo` 以管理员执行其他程序
- `su -` 切换用户身份
- `chmod` 修改权限
    - 三组二进制位 `rwx`
    - `chmod -R a+x,u+rw,g+r,o-w path`
- `chown -R `id -u`:`id -g` filename` 修改文件所有者和组

日志管理

- `cat` 查看文件
- `head -n N` 前 N 行
- `tail -n N` 后 N 行
- `less`
    - 按 j 向下
    - 按 k 向上
    - 按 f 向下翻屏
    - 按 b 向上翻屏
    - 按 g 到全文开头
    - 按 G 到全文结尾
    - 按 q 退出
- `sort` 排序
- `uniq` 去重, 依赖排序, 常跟在 sort 后面使用
- `awk '{print $N}'` 打印出相关列
- `wc` 字符(-c)、单词(-w)、行(-l)的计数
- `history | awk '{print $4}' | sort | uniq -c | sort -r | head -n 10`
- `watch -n 1.5 [command]` 动态查看执行结果

查找

- `grep`
  - 参数
    - `-i` 忽略大小写
    - `-I` 忽略二进制文件
    - `-r` 递归查找目录
    - `-n` 打印行号
    - `-c` 只显示匹配到的个数
    - `-l` 只显示匹配到的文件列表
    - `-o` 只显示匹配到的单词
    - `-v` 忽略制定的字段
    - `-E` 通过正则表达式匹配
    - `--include='*.py'` 仅包含 py 文件
    - `--exclude='*.js'` 不包含 js 文件
- `find DIR -name '*.xxx'` 找到目录下所有名字匹配的文件
  - 找出文件夹 /tmp/xyz/ 下所有的权限为 642, 大小在 10k 到 100k 之间的 log 文件
  - `find /tmp/xyz/ -perm 0642 -size +10k -size -100k -name '*.log'`
- `which` 精确查找当前可执行的命令
- `whereis` 查找所有匹配的命令

网络管理

- `ifconfig` 查看网卡状态
- `netstat -natp` 查看网络连接状态
- `ping -i 0.5`
- `lsof -i :[PORT]` 查看占用端口的程序
  - `lsof -i tcp` 查看所有 TCP 连接
  - `lsof -u abc` 查看用户 abc 打开的所有文件
  - `lsof -p 123` 查看 pid 为 123 的进程打开的所有文件
- `telnet [HOST] [PORT]` 查看远程主机网络连接状况
- `traceroute [HOST]` 路由追踪
- `dig [DOMAIN]` DNS 查询

下载

- `curl` 执行 HTTP 访问，也可用来下载
- `wget` 下载

远程登录

- `ssh [username@host]` 默认端口 22，其他端口使用 -p 参数
  - RSA key
  - `ssh-keygen` 创建自己的密钥对
  - 配置修改: `/etc/ssh/ssh_config`
  - 本地执行远程命令： `ssh username@host '[command]'`
  - 科学上网: `ssh -qTfnN -D 7070 [username@host]`
- SSH 服务端: `sshd`
  - 配置修改: `/etc/ssh/sshd_config`
  - 重启服务: `service ssh restart`
- `scp -P [PORT] filename username@host:/path/`
- `rsync -cvrP --exclude={.git,.venv} --delete`

压缩解压

- `tar`
  - 压缩： `tar -czf newfile.tgz files`
  - 解压： `tar -xzf file.tgz`
- `zip`
  - 压缩 `zip -r newfile.zip src-file1 src-file2 ...`
    - `-N` 指定压缩比
  - 解压 `unzip file.zip`

登录状态

- `uname -a`
- `hostname` 查看/修改主机名
- `w` 查看登陆者信息
- `who` 查看登陆者信息
- `whoami` 当前用户名
- `last` 最近登录记录

磁盘管理

- `du -hs` 查看文件或文件夹大小
- `df -h` 查看磁盘分区的占用情况
- `fdisk -l` 查看分区信息
- `dd` 以块的级别进行磁盘复制
  - `if` (input file) 输入文件
  - `of` (output file) 输出文件
  - `bs` (block size) 块大小 (单位: k, m, g)
  - `count` 块数量

    - `dd if=[src_file] of=[dst_file] bs=[size] count=[num]`
    - 从 iso 文件制作启动 U 盘: `dd if=/your_path/ubuntu.iso of=/dev/disk3 bs=1m`

安装

- `apt` debain 系 Linux 的程序安装
    - deb 安装包
    - 修改 apt 源
    - `apt update` 更新软件信息
    - `apt upgrade` 升级软件包
    - `apt search xxx` 查找相关软件包
    - `apt install xxx` 安装软件包
    - `apt remove xxx` 删除软件
- `yum` redhat 系 Linux 的程序安装
    - rpm 安装包
    - yum install xxx
- `make` 编译
    1. `./configure` 配置编译参数
    2. `make` 执行编译
    3. `make install` 安装编译文件到系统目录
    4. `make clean` 删除编译结果

文本处理工具

- `emacs`: 神的编辑器
- `vim`: 编辑器之神
    - esc 键，默认模式
        - `h, j, k, l` 光标左、下、上、右移动
        - `ctl + e` 向下滚动
        - `ctl + y` 向上滚动
        - `ctl + f` 向下翻屏
        - `ctl + b` 向上翻屏
        - `yy` 复制整行
        - `yw` 复制整行
        - `p` 粘贴到下一行
        - `P` 粘贴到下一行
        - `dd` 删除整行
        - `d3w` 向前删除3个单词
        - `7x` 删除7个字符
        - `u` 撤销
        - `ctl + r` 重做
        - `c3w` 剪切3个单词

        - `gg` 跳至文件首行
        - `shift + g` 跳至文件结尾
        - `shift + h` 跳至屏幕首行
        - `shift + m` 跳至屏幕中间
        - `shift + l` 跳至屏幕结尾
    - i 键，插入模式
    - : 键，命令模式
        - `23` 跳至文件的第 23 行
        - `%s/abc/123/g` 把文件中所有的 abc 替换成 123
        - `set nu` 打开行号
        - `set nonu` 关闭行号
        - `w` 保存
        - `q` 退出
        - `wq` 保存并退出
    - `ctl + v` 列编辑
    - `shift + v` 选中整列
    - `shift + >` 向右缩紧
    - `shift + <` 向左缩紧
    - 配置文件 .vimrc
    - 备注
        - https://coolshell.cn/articles/5426.html
        - http://www.oschina.net/question/615783_148433
        - 我的 vimrc https://raw.githubusercontent.com/seamile/rc.d/master/vimrc
- `sed` 流编辑器 (stream editor)
    - s 替换

```
sed -i 's/Python/Ruby/g' PythonZen.txt        # 替换并将修改写入原文件
sed '3,5s/^/# /g' PythonZen.txt               # 注释掉第2到5行
sed 's/b/____/2g' PythonZen.txt               # 替换每行第二个 `b`
sed '3,5s/^/# /g; s/b/____/2g' PythonZen.txt  # 组合操作
```

    - i 插入

```
sed '5i ====' PythonZen.txt                     # 在第 5 行前面插入一行
sed '/better/i ---> A new line' PythonZen.txt  # 在匹配到 `better` 的行前面插入一行
```

    - a 追加

```
sed '5a ====' PythonZen.txt                    # 在第 5 行前面插入一行
sed '/better/a ---> A new line' PythonZen.txt  # 在匹配到 `better` 的行后面追加一行
```

- c 行替换

```
sed '5c ---> A new line' PythonZen.txt         # 替换第 5 行
sed '/better/c ---> A new line' PythonZen.txt  # 把匹配到 `better` 的行替换成新行
```

- d 删除

```
sed '3,5d' PythonZen.txt                       # 删除第 3~5 行
sed '/better/d' PythonZen.txt                  # 删除包含 `better` 的行
```

- `awk`
  - 查找进程 ID: `ps ex | grep -i chrome | awk '{print $1}'`
  - 过滤网络信息: `netstat -nat | awk 'NR==1 || $6 ~ /(SYN|FIN|WAIT)/ {print $0}'`

环境变量

- `export` 设置一个全局环境变量
- `unset` 删除变量
- `$PATH` 可执行文件的存放目录
  - `export PATH=[your path]:$PATH`
- `$HOME` 家目录
- `$PWD` 当前目录

基础的的 bash 脚本语句

- `#!/bin/bash` 指明解释器
- `for...do...done`

```
for v in `ls ./`;
do
    echo $v
done
```

- `if...then...else...fi`

```
if [[ -d src ]];then
    echo "Yes"
else
    echo "No"
fi
```

- `echo`
- `printf`
- `seq` 相当于 Python 下的 range
  - `seq 1 5`

    ```
    for i in `seq 1 4`;
    do
        echo app-$i
    done
    ```

手册

- `man [command]` 操作手册

其他工具

- `git`
  - `git init` 初始化 .git 文件夹
  - `git add` 添加追踪
  - `git reset` 取消 add 状态
  - `git commit` 提交
  - `git status` 查看当前状态
  - `git checkout` 切换分支
  - `git branch` 分支管理
  - `git pull` 拉去远程代码
  - `git push` 将本地代码推送到远程库
  - `git diff` 差异对比
  - `git log` 查看提交历史
  - `~/.gitconfig`
  - 初始化新库

```
$ cd your_project_dir
$ git init
$ git add ./
$ git commit -m 'this is my first commit'
$ git remote add origin git@github.com:your_github_path.git
$ git push -u origin master
```

Python

- `pip`
- `virtualenv`
- `pyenv`

几个符号

- `|` 管道符: 把前面的输出结果作为后面命令的参数
- `>` 重定向: 把前面的输出结果重定向到指定的文件中
- `<` 重定向: 把前面的输出结果重定向到指定的文件中

简单的部署

运行一个django程序, 有数据库, nginx转发