TechNote

Lucas-Wye

2024 年 02 月 15 日

Contents:

CHAPTER 1

How to Ask Question

Click here

CHAPTER 2

How to use search engine

- Search engines are systems that enable users to search for documents on the World Wide Web.
- Popular examples include Yahoo!Search, Bing, Google, and Ask.com.

2.1 特殊符号

双引号：把搜索词放在双引号中，代表**完全匹配搜索**（顺序也必须完全匹配）

eg: “浙江大学 SCDA”

减号：搜索不包含减号后面的词的页面，使用这个指令时减号前面必须是**空格**，减号后面没有空格，紧跟着需要排除的词

eg: 浙江大学 -学院

星号：通配符

eg: 浙 * 大学

inurl：查找网址中包含指定字符的页面

eg: inurl:nice

inanchor：查找导入链接锚文字中包含搜索词的页面

eg: inanchor: 点击这里

intitle: 查找页面 title 中包含关键词的页面

eg: intitle: 查老师

filetype：查找特定格式文件

eg: filetype:pdf SCDA

site: 搜索某个域名下的所有子路径

eg: site: www.zju.edu.cn

2.2 快照功能

搜索引擎在收录网页时，对网页进行备份，存在自己的服务器缓存里，由于网页快照是存储在搜索引擎服务器中，所以查看网页快照的速度往往比直接访问网页要快

eg: 高斯分布 site:zh.wikipedia.org

2.3 特殊搜索

- 中文文献
  百度学术
  bing 学术
  谷歌学术
- 英文文献
  bing 学术
  谷歌学术
  Sci-hub
  IEEE
- 编程相关
  github
  medium
  stackoverflow

2.4 深度搜索

特殊的搜索工具可以搜索**深网**的内容

微信搜索 https://weixin.sogou.com

Archive 搜索引擎 http://archive.org

Wikihow https://zh.wikihow.com/搜索深网

more https://www.freebuf.com/news/137844.html

2.5 More

search.pdf
搜索引擎有哪些常用技巧
深网搜索引擎

CHAPTER 3

Git

- Git 是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。
- Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。
- Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

3.1 创建版本库

```
git init
git clone URL
```

3.2 修改和提交

```
git status
git diff FILE_NAME
git add FILE_NAME
git mv OLD_NMAE NEW_NAME
git rm FILE_NAME
git rm --cached FILE_NAME # 停止跟踪文件但不删除
git commit -m YOUR_COMMENT
git commit --amend
```

3.3 查看提交历史

```
git log
git reflog # 所有分支信息
git log -p FILE_NAME # 查看指定文件的提交历史
git blame FILE_NAME # 以列表方式查看指定文件的提交历史
```

3.4 撤销

```
git reset HEAD
git reset --hard HEAD
git checkout HEAD FILE_NAME
git revert COMMIT_ID
```

3.5 分支

```
# 查看所有远程分支
git branch -r
# 查看远程和本地所有分支
git branch -a
# 查看所有本地分支
git branch

# 切换分支
git checkout BRANCH

git branch NEW_BRANCH
git branch -d BRANCH
git branch -m OLD_NAME NEW_NAME

# 拉取远程分支并创建本地分支
#␣
↪(1)使用该方式会在本地新建分支x，并自动切换到该本地分支x，采用此种方法建立的本地分支会和远程分支建立
git checkout -b 本地分支名x origin/远程分支名x
#␣
↪(2)使用该方式会在本地新建分支x，但是不会自动切换到该本地分支x，需要手动checkout，采用此种方法建立的
git fetch origin 远程分支名x:本地分支名x

# 删除远程分支(1)
git branch -r -d origin/BRANCH_NAME
git push origin :BRANCH_NAME
# 删除远程分支(2)
git push origin --delete BRANCH_NAME
```

3.6 标签

```
git tag # 列出所有的本地标签
git tag TAG_NAME # 基于最新的提交创建标签
git tag -d TAG_NAME
```

3.7 合并与衍合

```
git merge BRANCH
git rebase BRANCH
```

3.8 远程操作

```
git remote -v
git remote show REMOTE # 查看指定远程版本库信息
git remote add REMOTE URL

git fetch REMOTE
git pull REMOTE BRANCH
git push REMOTE BRANCH
git push REMOTE:BRANCH:TAG
git push --tags
```

3.9 垃圾回收

Git 仓库越来越臃肿，大多数版本控制系统存储的是一组初始文件，以及每个文件随着时间的演进而逐步积累起来的差异；而 Git 则会把文件的每一个差异化版本都记录在案。这意味着，即使你只改动了某个文件的一行内容，Git 也会生成一个全新的对象来存储新的文件内容。

对象碎片：如果你改动了一个很大的文件，git 会为这个文件生成了一个很大的 Blob 对象

```
cd .git
du -ah  # 查看文件大小
git gc --prune=now # 垃圾回收
```

实际上，并不需要手动调用 gc 命令。每当碎片对象过多，或者你向远端服务器发起推送的时候，git 就会自动执行一次打包过程。

3.10 submodule

```
# 添加子仓库
git submodule add <仓库地址> <本地路径>
# 检出子仓库
git submodule init # 初始化本地配置文件
git submodule update # 检出父仓库列出的commit
## 或者
git submodule update --init --recursive
# 递归克隆
git clone <仓库地址> --recursive
```

3.11 合并多个 commit

```
commit THIRD_COMMIT_ID
        add third commit

commit SECOND_COMMIT_ID
        add second_commit

commit FIRST_COMMIT_ID
        add third commit
```

首先有 3 个 commit，需要将前两个 commit 合成一个

```
git rebase -i FIRST_COMMIT_ID
```

出现如下界面：

```
pick SECOND_COMMIT_ID add second_commit
pick THIRD_COMMIT_ID add third commit

...
```

修改成：

```
pick SECOND_COMMIT_ID add second_commit
squash THIRD_COMMIT_ID add third commit
```

DONE.

3.12 Git Server

```
sudo adduser git
su git
cd
# add ssh key
git init --bare [PROJECT_NAME].git
```

3.13 stash

- stash 是 Git 提供的一种机制，它可以将当前工作目录和暂存区的修改保存起来。当我们需要切换分支、执行 pull 操作或解决一些紧急 bug 时，stash 可以帮助我们保存当前的修改，避免丢失工作。
- 我们可以把 stash 看作是一个临时保存修改的容器，每次执行 stash 操作时，Git 将当前的修改保存到一个栈中，并将工作目录和暂存区恢复到最新的提交状态。stash 不会创建新的提交记录，它只是将修改暂时存储起来

执行 `git stash`，Git 会将当前的修改保存到 stash 中，并将工作目录和暂存区还原到最新的提交状态

执行 `git stash list`，可以查看 stash 列表，确认保存的 stash

执行 `git stash show stash@{0}`，查看某个 stash 中保存的具体修改内容

执行 `git stash apply stash@{0}`，将 0 所对应的内容应用到当前分支

执行 `git stash drop stash@{0}`，删除这个 stash

3.14 More

git 教程

CHAPTER 4

Kali

- Kali Linux 是基于 Debian 的 Linux 发行版，设计用于数字取证操作系统。由 Offensive Security Ltd 维护和资助。最先由 Offensive Security 的 Mati Aharoni 和 Devon Kearns 通过重写 BackTrack 来完成，BackTrack 是他们之前写的用于取证的 Linux 发行版。
- Kali Linux 预装了许多渗透测试软件，包括 nmap 、Wireshark 、John the Ripper，以及 Aircrack-ng；用户可通过硬盘、live CD 或 live USB 运行 Kali Linux。Kali Linux 既有 32 位和 64 位的镜像。可用于 x86 指令集。同时还有基于 ARM 架构的镜像，可用于树莓派和三星的 ARM Chromebook。

4.1 Install

```
# 从Docker安装
docker pull kalilinux/kali-linux-docker
# 运行
docker run -t -i kalilinux/kali-linux-docker /bin/bash
```

4.2 More

kali 官网

CHAPTER 5

tmux

- tmux 是一个优秀的终端复用软件，类似 GNU Screen，但来自于 OpenBSD，采用 BSD 授权
- 使用它最直观的好处就是，通过一个终端登录远程主机并运行 tmux 后，在其中可以开启多个控制台而无需再“浪费”多余的终端来连接这台远程主机

5.1 Install

```
# Ubuntu
sudo apt-get install tmux
# MacOS
brew install tmux
```

5.2 基本操作

```
# 新建会话
tmux # 新建一个无名称的会话
tmux new -s demo # 新建一个名称为demo的会话
# 进入之前会话
tmux a # 默认进入第一个会话
tmux a -t demo # 进入到名称为demo的会话
# 离开会话
[CTRL B] d
# 关闭会话
tmux kill-session -t demo # 关闭demo会话
tmux kill-server # 关闭服务器，所有的会话都将关闭
# 查看会话
tmux list-session # 查看所有会话
tmux ls # 查看所有会话，提倡使用简写形式
```

(续下页)

(接上页)

```
# 滚屏/cope mode
[CTRL B] [
# 设置滚屏vi快捷键
echo "setw -g mode-keys vi" >  ~/.tmux.conf
tmux source-file ~/.tmux.conf
# copy
[CTRL B] [  -> [Space] -> Select -> [Enter]
# paste
[CTRL B] ]

# 新建窗口
[CRTL B] c
# 切换窗口
[CRTL B] n

# 切换pane
[CTRL B] Up|Down|Left|Right
# 垂直分屏
[CTRL B] %
# 水平分屏
[CTRL B] "
```

5.3 More

tmux 使用手册

CHAPTER 6

Linux

- Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 Unix 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的 Unix 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

6.1 Control Key

```
[CTRL]U           cancel line
[CTRL]C           cancel operation
[CTRL]S           pause display
[CTRL]Q           restart display
[CTRL]A           光标移到行首
[CTRL]E           光标移到行末
[CTRL]K           清除至当前行尾
[CTRL]V           treat following control character as normal character
[Option]方向键      以单词为单位移动
```

6.2 User

```
sudo adduser USERNAME
# 添加root权限
sudo usermod -g sudo USERNAME
# change password
passwd
# delete user
sudo userdel -r USERNAME
```

6.3 Infomation

```
who
who am i
whoami
env
alias
man
```

6.4 File Maintenance

```
# r = 4, w = 2, x = 1
chmod
umask # set in startup files for the account to masks out permissions, umask numbers␣
↪added to desired permission number equals 7.
chgrp # change the group of the file
chown # change the owner of a file

# 查看当前目录文件大小
# (1)列出当前目录下每个文件的大小，同时也会给出当前目录下所有文件大小总和
ls -lht

# (2)列出当前文件夹下所有文件对应的大小
du -sh PATH

# 删除文件中的空行
cat YOUR_FILE | sed -e '/^$/d'

# conditions
-r return true (1) if it exists and is readable, otherwise return false (0)
-w true if it exists and is writable
-x true if it exists and is executable
-f true if it exists and is a regular file (or for csh, exists and is not a directory)
-d true if it exists and is a directory
-e true if the file exists
-o true if the user owns the file
-z true if the file has zero length (empty)

# 对Exfat文件系统支持
sudo apt install exfat-utils

# 打包
tar -cvf  YOUR_FILE.tar YOUR_FILE # 仅打包
tar -zcvf YOUR_FILE.tar.gz YOUR_FILE # gzip压缩
tar -jcvf YOUR_FILE.tar.bz2 YOUR_FILE # bzip2压缩

# 查看文件
tar -tvf YOUR_FILE.tar
tar -ztvf YOUR_FILE.tar.gz
tar -jtvf YOUR_FILE.tar.bz2

# 解包
tar -xvf YOUR_FILE.tar
tar -zxvf YOUR_FILE.tar.gz
```

(续下页)

(接上页)

```
tar -jxvf YOUR_FILE.tar.bz2
```

6.5 find and search

```
# 查找24小时内修改过的文件
find ./ -mtime 0
# 查找当前目录及子目录中的.c文件
find . -name "*.c"
# 查找当前目录符合条件的文件内容
grep -nHR "STRING" .
# grep不匹配二进制文件
grep --binary-files=without-match
```

6.6 process

```
ps
ps -ef
kill -9 PID
```

6.7 Bash executes order

```
# login shell executes order:
/etc/profile
~/.bash/_profile
~/.bash_login
~/.profile

# non-login shell executes:
/etc/bashrc
~/.bashrc
```

6.8 History

```
history
!598 # 执行第598条命令
sudo !! # 以root执行上一条命令
history  | awk '{a[$2]++}END{for(i in a){print a[i] " " i}}' | sort -rn | head #␣
↪统计情况
```

6.9 CPU

```
# 总核数 = 物理CPU个数 x 每颗物理CPU的核数
# 总逻辑CPU数 = 物理CPU个数 x 每颗物理CPU的核数 x 超线程数

# 物理CPU个数
cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l

# 每个物理CPU中core的个数(即核数)
cat /proc/cpuinfo| grep "cpu cores"| uniq

# 逻辑CPU的个数
cat /proc/cpuinfo| grep "processor"| wc -l

# CPU型号
cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

# CPU的负载, 返回1、5、15分钟内的负载情况
uptime
```

6.10 内存

```
cat /proc/meminfo
free
```

6.11 磁盘

```
# 硬盘信息
fdisk -l
# 查看磁盘IO的性能
iostat -x 10
# 挂载硬盘到某个文件夹
sudo mount /dev/sda YOUR_PATH
# 查看硬盘挂载信息
df -h
# 取消挂载
sudo umount YOUR_PATH
```

6.12 Ubuntu 开机自动挂载 Windows 硬盘分区

```
查看分区信息
sudo fdisk -l
查看磁盘类型
sudo blkid
```

输出

```
Device          Boot      Start        End    Sectors  Size Id Type
/dev/nvme0n1p1  *          2048    1187839    1185792  579M  7 HPFS/NTFS/exFAT
/dev/nvme0n1p2          1187840  210903039  209715200  100G  7 HPFS/NTFS/exFAT
/dev/nvme0n1p3        210903040  420618239  209715200  100G  7 HPFS/NTFS/exFAT
/dev/nvme0n1p4        420620286  500117503   79497218 37.9G  5 Extended
/dev/nvme0n1p5        420620288  421595135     974848  476M 83 Linux
/dev/nvme0n1p6        421597184  450891775   29294592   14G 83 Linux
/dev/nvme0n1p7        450893824  500117503   49223680 23.5G 83 Linux
```

```
修改配置文件
sudo vim /etc/fstab

# for Windows 10 C:/
/dev/nvme0n1p2 /home/usrname/Windows_Disks/C ntfs defaults 0 0

# for Windows 10 D:/
/dev/nvme0n1p3 /home/usrname/Windows_Disks/D ntfs defaults 0 0

挂载新添加的分区
sudo mount -a
```

6.13 删除 ppa 源

```
cd /etc/apt/sources.list.d/ # 找到关于源的文件,删除即可
```

6.14 字体

```
# 安装Windows字体
sudo cp [Windows-Fonts] /usr/share/fonts/Windows-Fonts
sudo mkfontscale
sudo mkfontdir
fc-cache

# 查看中文字体
fc-list:lang=zh-cn
```

6.15 I/O Redirection and Piping

```
# stdin: 0, stdout: 1, stderr: 2
|                       管道
>                       stdout重定向到file
>>                      stdout重定向到file(不覆盖)
<                       stdin从file重定向
tee                     复制stdout
>/dev/null              直接扔掉stdout
1>FILE_1 2>FILE_2         stdout to FILE_1, stderr to FILE_2
>FILE 2>&1              redirect stdout and stderr to FILE
2>&1 | tee              将stderr和stdout输出到文件的同时在屏幕上输出
```

6.16 开机进入命令行

```
sudo systemctl set-default multi.user # 进入命令行
sudo systemctl set-default graph...   # 进入图形界面
```

6.17 命令行切换回 GUI

```
startx
sudo service gdm3 restart
```

6.18 Ubuntu 设置窗口键在左侧

```
gsettings set org.gnome.desktop.wm.preferences button-layout 'close,minimize,maximize:
↪'
```

6.19 生成强密码

```
openssl rand -base64 NUMBER
```

6.20 terminal output to clip

- Windows: `clip`
- MacOS: `pbcopy`, `pbpaste`
- Linux: `xsel`

6.21 ssh

```
# 安装 SSH(Secure Shell) 服务以提供远程管理服务
sudo apt install openssh-server

# 启动ssh服务
/etc/init.d/ssh start
sudo service ssh start

# 检测是否已启动
ps -e | grep ssh

## SSH远程登录
ssh username@IP_ADDR

# 将文件/文件夹从远程机下载到本地(scp)
```

(续下页)

(接上页)

```
scp -r username@IP_ADDR:/home/username/remotefile.txt .

# 设置公钥登录
# (1)复制本地的公钥
cat ~/.ssh/id_rsa.pub
# (2)在远程机器上写入复制的公钥
vim ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
# (3)远程机器授权公钥登录
sudo echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config
# (4)重启ssh服务
sudo systemctl restart sshd.service
# Or
ssh-copy-id -i Public_Key_File Remote_Server

# .ssh/config example
Host {HOSTNAME}
  HostName {IP}
  User {Username}

ssh HOSTNAME

# SSH for data transfer
ssh -qTfnN -D PORT SERVER
```

6.22 Mac OS Theme for Ubuntu

```
sudo apt install gnome-tweak-tool gnome-shell-extensions

git clone --depth=1 https://github.com/vinceliuice/Sierra-gtk-theme
cd Sierra-gtk-theme
./install.sh

git clone --depth=1 https://github.com/USBA/Cupertino-iCons
sudo cp -r Cupertino-iCons /usr/share/icons
```

安装完成后，打开 tweak
设置 Appearance->Themes->Applications 为 Sierra-light
设置 Appearance->Themes->Icons 为 Cupertino-iCons

6.23 More

A good introduction to Linux

CHAPTER 7

Docker

- Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 Linux 或 Windows 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

7.1 Install

```
# Ubuntu
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# 建立docker组
sudo groupadd docker
# 将当前用户加入docker组
sudo usermod -aG docker $USER

# 卸载本机所有的镜像、容器、卷以及配置文件
sudo rm -rf /var/lib/docker
```

7.2 Basic

Pull

```
docker pull [user name]/[repo name]:[tag name]
```

Run

- -i 以交互模式运行容器，通常与 -t 同时使用
- -t 为容器重新分配一个伪输入终端，通常与 -i 同时使用
- -v 绑定一个卷，格式为：本机绑定目录: 容器内部绑定目录

- -d 后台运行容器，并返回容器 ID
- -P 随机端口映射，容器内部端口随机映射到主机的高端口
- -p 指定端口映射，格式为：主机 (宿主) 端口: 容器端口
- -a 指定标准输入输出内容类型，可选 STDIN/STDOUT/STDERR 三项
- –name 对所新建容器进行命名
- –rm 容器终止后，自动删除容器文件

Others

```
# 列出容器
docker container ls -a
# 查看容器ID
docker ps -a
# 重新启动已被终止的指定容器
docker start [CONTAINER ID]
# 终止容器
docker stop [CONTAINER NAME/ID]
# 若是利用 -it 在容器内部进行操作，仅需输入 exit 即可
# 删除容器
docker kill [CONTAINER NAME/ID]
# 将所有容器删除
docker container prune
# 列出镜像
docker images
# 重命名镜像
docker tag [old REPOSITORY]:[old TAG] or [IMAGE ID] [new REPOSITORY]:[new TAG]
# 删除镜像
docker rmi [IMAGE]
# 存出镜像
docker save -o [xx.tar] [REPOSITORY]:[TAG]
# 载入镜像
sudo docker load --input [镜像文件]
# 更新镜像
docker commit [OPTIONS] [IMAGE ID] [new REPOSITORY]:[new TAG]
# -m: 提交的描述信息
# -a: 指定镜像作者
```

7.3 More

Docker 教程
Docker 学习笔记

CHAPTER 8

Zotero

Zotero is a free, easy-to-use tool to help you collect, organize, cite, and share research.

8.1 Tips

Requirements	Operations
查看文献条目所属的分类	选中该条目后按住 Ctrl/Option/Alt，该文献所在的文件夹就会高亮为黄色
在文献集（Collections）之间移动文献	选中待移动的文献，然后根据你的操作系统按下相应快捷键，macOS：`Cmd` Windows/Linux：`Shift`，将该文献拖拽到其他文献集即可
快速查看近期添加的文献	右键单击 **My Library**，选择 **New Saved Search**，进行具体的设置

8.2 Plug

8.2.1 Better Bibtex

- 设置格式

`Zotero Preferences` -> `Better Bibtex` -> `Citation Keys` 中，修改 `citation key format` 为

```
[auth:lower][year][veryshorttitle:lower]
```

8.2.2 ZotFile

- 修改附件的命名格式

`Zotero Tools` -> `ZotFile Preferences` -> `Renaming` 中，修改格式为 `{%y_}{%t_}{%a}`

8.2.3 Zotero Scholar Citations

8.3 More

Zotero Website

CHAPTER 9

Network

- A computer network is a digital telecommunications network which allows nodes to share resources. In computer networks, computing devices exchange data with each other using connections (data links) between nodes.

9.1 Commands

```
ping
domainname
hostname

cat /etc/hosts # ip address
sudo systemctl restart NetworkManager # hosts生效

cat /etc/resolv.conf # dns server
ip # TCP/IP interface configuration and routing utility
ifconfig # configure a network interface
route # show / manipulate the IP routing table
netstat # show network status (network connections, routing tables, interface␣
↪statistics, masquerade connections, and multicast memberships)

sudo ping ip地址 -i 0.01 -s 65500 # 每0.01秒给ip地址对应的机器发送65500字节的数据包
```

9.2 Useful Remote Connection Utilities

- **ftp** [options] host, transfer file(s) using file transfer protocol
- **telnet** [host [port]], communicate with host using telnet protocol
- **ssh**, remote login or remote execution using secure shell
- **rcp/scp**, remotely copy files from this machine to another machine
- **rsync**, smartly copy files over network after checking contents
- **curl**, transfer a URL via HTTP, FTP, IMAP, etc
- **wget**, download files over the Internet via HTTP or FTP
- **lynx/links**, text-mode (mini) web browser

9.2.1 aria2

(1)Install

```
# Ubuntu
sudo apt-get install aria2
# CentOS
yum install aria2
```

(2)Usage

```
# 在命令后附加地址即可
aria2c "url"
# 分段下载，利用 aria2 的分段下载功能可以加快文件的下载速度
# 使用 2 个连接来下载该文件，s的参数值介于 1~5 之间
aria2c -s 2 "url"

# 断点续传，在命令中使用 c 选项可以断点续传文件
aria2c -c "url"
```

9.2.2 curl

- client URL tool

(1) 不带有任何参数时，发出 GET 请求

```
curl https://www.example.com
```

(2)`-A` 指定 User-Agent，默认用户代理字符串是 curl/[version]

```
curl -A 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like␣
↪Gecko) Chrome/76.0.3809.100 Safari/537.36' https://google.com
```

(3)`-b` 参数用来向服务器发送 Cookie

```
# 生成一个标头Cookie: foo=bar，向服务器发送一个名为foo、值为bar的 Cookie
curl -b 'foo=bar' https://google.com
```

(4)`-d` 参数用于发送 POST 请求的数据体

```
curl -d'login=emma&password=123'-X POST https://google.com/login
# `--data-urlencode` 等同于 `-d`
# 发送 POST 请求的数据体，区别在于会自动将发送的数据进行 URL 编码
```

(5)-G 参数用来构造 URL 的查询字符串

```
# 实际请求的 URL 为https://google.com/search?q=kitties&count=20
curl -G -d 'q=kitties' -d 'count=20' https://google.com/search
```

(6)-H 参数添加 HTTP 请求的标头

```
curl -H 'Accept-Language: en-US' https://google.com
curl -d '{"login": "emma", "pass": "123"}' -H 'Content-Type: application/json' https:/
↪/google.com/login
```

(7) -i 参数打印出服务器回应的 HTTP 标头

```
# 先输出服务器回应的标头，然后空一行，再输出网页的源码
curl -i https://www.example.com
```

(8)-o 参数将服务器的回应保存成文件，等同于 wget 命令

```
curl -o example.html https://www.example.com
# `-O` 参数将服务器回应保存成文件，并将 URL 的最后部分当作文件名。
curl -O https://www.example.com/foo/bar.html
# 通过添加 `-C` 继续对该文件进行下载，已经下载过的文件不会被重新下载
curl -C -O http://www.gnu.org/software/gettext/manual/gettext.html
```

(9) 使用 wget 或者 curl 下载 github release 文件

```
curl -LJO GITHUB_RELEASE_LINK
wget --no-check-certificate --content-disposition GITHUB_RELEASE_LINK
```

9.2.3 terminal 设置代理

```
# MacOS & Linux
export http_proxy=http://127.0.0.1:PORT # PORT替换成具体的端口
export https_proxy=http://127.0.0.1:PORT
# Windows
set https_proxy="http://127.0.0.1:PORT"
set http_proxy="http://127.0.0.1:PORT"
```

9.2.4 nmcli

```
# connect to l2tp
nmcli con up {L2TP_VPN_NAME} --ask
```

9.2.5 Windows VPN 连接

```
rasdial VPN_NAME USERNAME PASSWORD
```

9.3 More

CHAPTER 10

Vi Editor

- 所有的 Unix Like 系统都会内建 Vi 文书编辑器，其他的文书编辑器则不一定会存在。但是目前我们使用比较多的是 Vim 编辑器。
- Vim 具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。
- Vim 是从 Vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。简单的来说，Vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。

10.1 File Operation

:wq	存盘退出
:q!	退出不保存
:saveas [path/to/file]	另存为 [path/to/file]
:qa!	强行退出所有的正在编辑的文件
:bn 和:bp	切换下一个或上一个文件
:w !sudo tee %	以 sudo 保存正在编辑的文件
:n	move to next file
:rew	回到第一个文件
:edit [Filename]	打开新文件

10.2 Cursor movement

hjkl	move
/[PATTERN]	搜索
?[PATTERN]	搜索
/\<[PATTERN]\>	精确匹配搜索
%	匹配括号移动
f	搜索并移动到某个字符前
t	到某个字符前的第一个字符
:N	到第 N 行
gg	到第一行
G	到最后一行
[n]G	go to last line or line [n]
w/W	到下一个单词的开头
e/E	到下一个单词的结尾
b	到上一个单词的开头
0	beginning of current line
$	end of current line
^	beginning of text on current line
[CTRL] F/B	move screen
[CTRL] D/U	move half screen
zz	将当前行放置于屏幕中间
zt	将当前行放置于屏幕顶端
zb	将当前行放置于屏幕底端

10.3 Inserting text

i	前插入
a	后插入
I	insert text at beginning of line
A	append text at end of line
o	在当前行后插入一个新行
O	在当前行前插入一个新行

10.4 Deleting text

d	删除
D	删除当前行光标后所有内容
x	删当前光标的字符
X	删当前光标前的字符

10.5 Changing commands

u	undo
[CTRL] r	redo
.	repeat last operation
p	后粘贴
P	前粘贴
y	复制
s/S	substitute
~	change case of character
J	join current line and next line

10.6 Split windows

:split/vsplit	创建分屏
[CTRL] w + 方向	切换分屏

10.7 Command line mode

:![cmd]	暂时退出命令行执行 cmd
:set all	display all option settings
:[Addr/%]s/old expr/new string/[g]	替换当前行/Addr/%(文件内所有) 的 old expr 为 new string,[g] 全局替换，否则只替换第一个 |
[CTRL] p/n	自动补齐
[CTRL] +/-	改变尺寸
v	visual 模式
V	v-line 模式
[CTRL] v	v-block 模式
:normal [Command]	可视化模式下执行命令
qa	录制宏
ci + “	删除引号之中的内容
tabe	打开新的标签页
+/-tabnext	切换标签页

10.8 Regular expression

?	match any single character at the indicated position
*	match any string of zero or more characters
[abc…]	match any of the enclosed characters
[a-e]	match any characters in the range a,b,c,d,e
[!def]	match any characters not one of the enclosed characters, sh/bash
{abc,bcd,cde}	match any set of characters separated by comma (,) (no spaces), bash/csh
~	home directory of the current user, bash/csh
~ user	home directory of the specified user, bash/csh
.	match any single character except newline
[^abc]	match any character NOT in the enclosed set
^exp	regular expression must start at the beginning of the line
exp$	regular expression must end at the end of the line
\	treat the next character literally 转义字符
xy*z	xy 开头，z 结尾的字符串

10.9 Plug

https://github.com/junegunn/vim-plug

10.10 .vimrc

vimrc

10.11 NeoVim

使用 Vim 配置文件

```
ln -s ~/.vim ~/.config/nvim
ln -s ~/.vimrc ~/.config/nvim/init.vim
```

10.12 More

Vi/Vim 教程
简明 VIM 练级攻略

CHAPTER 11

Emacs

- GNU Emacs, An extensible, customizable, free/libre text editor —and more.
- At its core is an interpreter for Emacs Lisp, a dialect of the Lisp programming language with extensions to support text editing.

11.1 Cursor movement

C-v	Move forward one screenful
M-v	Move backward one screenful
C-l	Clear screen and redisplay all the text
C-n	Move next line
C-p	Move previous line
C-f	Move forward a character
C-b	Move backward a character
M-f	Move forward a word
M-b	Move backward a word
C-a	Move to beginning of line
C-e	Move to end of line
M-a	Move back to beginning of sentence
M-e	Move forward to end of sentence
M-<	Move to the beginning of file
M->	Move to the end of file
C-u NUM	(Prefix) repeat NUM times
C-g	stop/cancel current execution

11.2 Insert and Delete

DEL	delete the character just before the cursor
C-d	delete the next character after the cursor
M-DEL	kill the word immediately before the cursor
M-d	kill the next word after the cursor
C-k	kill from the cursor position to end of line
M-k	kill to the end of the current sentence
C-SPACE + [Move Cursor]	select text
C-y	yank the more recent killed text back
M-y	replace the yanked text with the previous kill
C-/, C-_, C-x u	Undo

11.3 File and Buffer

C-x C-f	find a file
C-x C-s	save the file
C-x C-b	list buffers
C-x b	switch buffer
C-x s	save some buffers
C-x C-c	quit the Emacs session
C-z	exit Emacs temporarily
M-x	prefix of some commands

11.4 Search

C-s	search forward
C-r	search backward

11.5 Windows

C-x 1	get rid of extra windows and go back to basic one-window editing
C-u 0 C-l	
C-h k C-f	
C-x 2	split the screen into two windows
C-M-v	scroll the bottom window
C-v o	move the cursor to the other window

11.6 配置镜像源

清华源

11.7 More

Spacemas

CHAPTER 12

GCC

- GCC 编译器也称为 Linux GCC 命令，它有很多选项。GCC 编译器是 Linux 下最常用的编译器。

12.1 Install

```
# MacOS安装Xcode工具链
xcode-select --install
# Ubuntu
sudo apt install gcc
```

12.2 Basic

```
gcc -c filename.c # compile only, produce .o
gcc -g # compile for debugging
gcc -o filename.o #
gcc -O 1,2,3,4,s,fast # for optimization level
gcc -Ipathname
gcc -Dsymbol # define preprocessor symbol
gcc -Ldirectory # add directory to the library search path
gcc -lxyz # link with library libxyz.a or libxyz.so
```

12.3 gdb

```
gdb BINARY_FILE
list
br 8 # breakpoint in line 8
run
print value
next
where
help
quit
```

12.4 multiple gcc version in one Ubuntu machine

use `update-alternatives` to manage them.

```
sudo update-alternatives --config gcc
```

12.5 make

(1)Predefined Macros

- AS - assembler (as)
- CC - C compiler command (cc)
- FC - Fortran compiler command (fc)
- CPP - C++ preprocessing command ($(CC) -E)
- CXX - C++ compiler command (g++)
- CFLAGS - C compiler option flags (e.g. -g)
- FFLAGS - Fortran compiler option flags (e.g. -g)
- LDFLAGS - Linking option flags (e.g. –L /usr/share/lib)
- LDLIBS –Linking libraries (e.g. -lm)

(2)Special Internal Macros

```
$* # The basename of the current target
$< # The name of a dependency file, as we see on last page
$@ # The name of the current target.
$? # The list of dependencies that are newer than the target.
```

12.6 More

more of gdb
Makefile example

CHAPTER 13

OpenMP

- OpenMP 是由 OpenMP Architecture Review Board 牵头提出的，并已被广泛接受的，用于共享内存并行系统的多线程程序设计的一套编译指令 (Compiler Directive)。

13.1 Introduction

See detail at wiki.

13.2 OpenMP Programming Example

Here is a C program using OpenMP.

```c
#include<stdio.h>
#include<omp.h>

int main(void) {
    const int n = 10;
    int arr[n];

    #pragma omp parallel for
    for(int i = 0;i < n;i++) {
        arr[i] = i;
        printf("%d\n",i);
    }

    for(int j = 0;j < n;j++){
        printf("%d\n",arr[j]);
    }

    return 0;
}
```

13.3 More Example

OpenMP 并行开发（C++）
OpenMP 并行程序设计（二）

CHAPTER 14

Java

- Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的高级程序设计语言。Java 可运行于多个平台，如 Windows, Mac OS，及其他多种 UNIX 版本的系统。

14.1 Install

JDK
IntelliJ IDEA

```
# 配置环境变量
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_144  # 这里换成自己的jdk目录
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
```

14.2 Usage of adb(Android Debug)

```
# 卸载系统软件
adb shell
pm list package
pm uninstall -k --user 0 package_name
```

CHAPTER 15

LaTeX

- LaTeX（音译“拉泰赫”）是一种基于 TEX 的排版系统，由美国计算机学家莱斯利 · 兰伯特（Leslie Lamport）在 20 世纪 80 年代初期开发。利用这种格式，即使使用者没有排版和程序设计的知识也可以充分发挥由 TeX 所提供的强大功能，能在几天，甚至几小时内生成很多具有书籍质量的印刷品。
- 对于生成复杂表格和数学公式，这一点表现得尤为突出。因此它非常适用于生成高印刷质量的科技和数学类文档。这个系统同样适用于生成从简单的信件到完整书籍的所有其他种类的文档。

15.1 Install

从清华镜像源下载对应操作系统的 Texlive 软件包

```
# 安装
sudo ./install-tl
# 设置环境变量
# LaTeX
export TEX_HOME=/usr/local/texlive/2019
export PATH=$PATH:$TEX_HOME/bin/x86_64-linux
export INFOPATH=$INFOPATH:$TEX_HOME/texmf-dist/doc/info
export MANPATH=$MANPATH:$TEX_HOME/texmf-dist/doc/man
```

15.2 安装 Windows 字体

```
# 创建 win 下字体专用文件夹
sudo mkdir /usr/share/fonts/winfonts

# 复制windows上的字体到/usr
sudo cp your_winfonts_dir /usr/share/fonts/winfonts

# 进入字体文件夹
```

(续下页)

(接上页)

```
cd /usr/share/fonts/winfonts

# 修改访问权限
sudo chmod 744 *

# 回到主目录
cd ~

# 更新字体信息
sudo mkfontscale
sudo mkfontdir
sudo fc-cache -f -v
```

15.3 Package

- Algorithm2e

15.4 More

LaTeX 模板
LaTeX 开源小屋
Stackexchange
LaTeX Introduction
How I’m able to take notes in mathematics lectures using LaTeX and Vim

CHAPTER 16

Python

- Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。
- Python 由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。像 Perl 语言一样, Python 源代码同样遵循 GPL(GNU General Public License) 协议。

16.1 Install

从清华镜像源下载对应平台的 anaconda/miniconda 安装即可。

16.2 conda 创建 python 虚拟环境

```
# 查看当前存在哪些虚拟环境
conda env list
## or
conda info -e
# 创建虚拟环境
conda create -n your_env_name python=X.X  # 版本选择：2.7、3.6等
# 激活环境
conda activate your_env_name
# 关闭虚拟环境
conda deactivate
# 安装package到your_env_name
conda install -n your_env_name package_name
# 删除环境中的某个包
conda remove --name your_env_name package_name
# 删除虚拟环境
conda remove -n your_env_name --all
```

16.3 常用包安装方法

```
# tensorflow
conda install tensorflow
# pytorch(CPU)
conda install pytorch torchvision cpuonly -c pytorch
```

16.3.1 conda 导出环境依赖

```
conda list -e > requirements.txt
```

16.3.2 Jupyter notebook

访问远程服务器的 notebook 配置：

```
# 生成一个 notebook 配置文件
jupyter notebook --generate-config
# 生成密码
jupyter notebook password
```

修改配置文件 ~/.jupyter/jupyter_notebook_config.py

```
c.NotebookApp.ip = '*' # 允许任何IP访问
c.NotebookApp.password = u'sha:...' # 密码的哈希值
c.NotebookApp.open_browser = False
c.NotebookApp.port = 8888 # 自行指定一个端口
```

16.3.3 PyPy

https://www.pypy.org
https://github.com/rvianello/conda-pypy

16.3.4 Cython

https://cython.org

16.3.5 镜像源

cat ~/.condarc

```
channels:
 - defaults
 - ric/channel/pypy

show_channel_urls: true

default_channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
```

(续下页)

(接上页)

```
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r

custom_channels:
 conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
 pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud

envs_dirs:
 - ~/pypy-envs
```

16.4 pip

```
# 导入依赖包
pip install -r requirements.txt
# 只导出项目依赖包
pip install pipreqs
pipreqs ./
# 导出依赖包
pip freeze > requirements.txt
# 离线下载
pip download -d [DIR] -r requirements.txt
# 离线安装
pip install --no-index --find-links=[DIR] -r requirements.txt
```

16.4.1 镜像源

cat ~/.pip/pip.conf

```
[global]
index-url = http://mirrors.aliyun.com/pypi/simple/
[install]
trusted-host = mirrors.aliyun.com
```

16.5 More

ipynb 文件在线查看工具

CHAPTER 17

Golang

- Go 是一个开源的编程语言，它能让构造简单、可靠且高效的软件变得容易。
- Go 是从 2007 年末由 Robert Griesemer, Rob Pike, Ken Thompson 主持开发，后来还加入了 Ian Lance Taylor, Russ Cox 等人，并最终于 2009 年 11 月开源，在 2012 年早些时候发布了 Go 1 稳定版本。
- 现在 Go 的开发已经是完全开放的，并且拥有一个活跃的社区。

17.1 Install

从官网下载对应平台的编译器

17.2 设置环境变量

```
# Golang
export GO=... # 安装位置
export PATH=$PATH:$GO/bin

export GOPATH=... # 指定一个本地位置
export PATH=$PATH:$GOPATH/bin
```

17.3 More

go 教程
go web 教程

CHAPTER 18

Matlab

Click here

CHAPTER 19

Indices and tables

- genindex
- modindex
- search