linux课堂

vim

- :sp

> 打开新的窗口哦

- :qa

> quit all

- w

> move by word

- e

> move to end of word

- 0

> move to the begin of line

- ^U, ^D

> quickly move up or down

- f[c]

> move to the first caracter

- dw

> delete the word

- dd

> delete the line

- cc

> delete the line but different from dd

- u

> undo the change

- ^r

> recover

- y

> copy

- p

> copy

- 4j

> move up 4 lines

- 4k

> move down 4 lines

- v

> go into visual mod

- %

> jump to the match

- d()

> delete the string in the ()

- /[]

> find something

- $

> move to the end of the line

- gg

> back to the start of file

- vim ~/.vimrc

> change the setting of vim

tar 简单用法

- `tar zcf name.tar.gz name --exclude=location`

> 排除location的文件

- `tar -xzvf name.tar.gz`

> 解压文件

- `tar -tzvf name.tar.gz`

> 列出压缩文件内容

- `tar -czvf name.tar.gz. name.c`

> 压缩name.c 为name.tar.gz

常用命令

- echo

> 用于输出字符串

- which

> 找到命令的位置，如which echo, which yums

- pwd

> 显示当前路径

- cd

> change the dir

- ls

> show all the files under the path

- cd -

> go back to last dir

- ls -l

> more information in ls

- mv [a] [b]

> move something to something

- man [command]

> see how to use command

- >

> mean the stream into
>
> `cat < A >> B`mean copy A to B twice
>
> > > mean add to, > mean refer to

- tail

> show the file
>
> `tail -n + 20 A`

> 菜鸟教程
>
> `ls -l \ | tail -n1`

- find

> `find . -name "*.c"` 当前目录下后缀为.c的
>
> `find . -type f`
>
> `find /var/log -type f -mtime +7 -ok rm{}\`
>
> 菜鸟教程

- grep

> to find the string in the file

- xdg-open

> open the file in the OS

- `scp -r [local file] root@[ip]:[remote file]`

> send local file to remote file

- `scp root@[ip]:[remote file] [local file]`

> get remote file to local file

shell脚本

- foo=bar
- "$foo"

> bar

- '$foo'

> $foo

- 在shell脚本中, "$1"表示一个参数，相当于argv[0]
- ex1

> `mcd(){}` mch.sh
>
> `source mch.sh`此时mcd被加入bash
>
> 使用mcd即可使用该函数

- $_

> last argv

- $?

> The wrong code, you can echo $? to see, zero means right

- $$

> pid

- $@

> when you do not know how many arg

- ls *.sh

> show all the sh file

- ls project?

> show all file like project1 ..

- tounch

> 菜鸟教程
>
> 改变属性，也可以用于创建文件

- !/usr/bin/env python

- shellcheck

> to check the sh file

- tldr

> like man

- locate

> find all the thing

- updatedb

> update the file name in the db of OS

- grep -R

> search in this dir of something

- histroy

> print all the command history

- cat A | fzf

make

> 视频

ex1

- `touch main.c tool1.c tool2.c tool1.h tool2.h`

- main.c

```main函数

```
#ifdef TOOL1_H_

#define TOOL1_H_

#endif

#include<stdio.h>

main(){}
```

```

- makefile

```

OBJS=main.o tool1.o tool2.o #用以代替下方的内容 CC=gcc #使用$()来获取 CFLAGS+=-c -Wall -g#当需要字符串组合时使用+

```
mytool:main.o tool1.o tool2.o
    gcc main.o tool1.o tool2.o

main.o:main.c
    gcc main.c -c Wall -g -o main.c

tool1.o:tool1.c
    gcc tool1.c -c Wall -g -o tool1.c

tool2.o:tool2.c
    gcc tool2.c -c Wall -g -o tool2.c

clean:
    rm *.o mytool -rf
```

```

- include 参与所欲不需要依赖h文件

- 执行clean 使用`make clean`

- $^ 表示本部分被依赖的部分， $@表示本部分被生成的部分

- `%.o:%.c` `$(CC) $^ $(CFLAGS) -o $@`

> %的用法

job

- nohup &

> 后台运行

- bg %1

> 重新唤起

- fg %1

> 召唤至前台

- kill

> 发起信号

- -HUP

> 挂起

- -KILL

> 杀死

- -STOP

> 暂停

- htop

> 监视器

- alias

> make a short word compared to long command string
>
> `alias ll = "ls -lah"`

- dotfiles

> can search it on github and change it.

ssh

- scp sth root@ip:/
- rsync -avP . root@ip:/

tmux

> http://www.ruanyifeng.com/blog/2019/10/tmux.html
>
> similar to screen

- tmux (new - s name)

> 进入一个窗口
>
> 括号里新建名字

- ctrl d

> exit

- tmux a

> show all the sesion

- tmux detach

> 与ctrl d一致

- tmux attach -t [0]/name

> 杀死会话

- tmux kill-session -t [0]/name

> 杀死会话

- tmux split-window

> 划分上下两个

- tmux split-window -h

> 划分左右

git

- git config --global user.email "you@example.com"
- git config --global user.name "Your Name"

basic

- git init

> init git hub

- git status

> show the history commit

- git add [file]

> add new file into git "new file"

- git commit

> Then it will ask u to type the name of commit, after that will return the hash of node/

- git log

> `git log --all --gragph --decorate [--oneline]`
>
> > more powerful one
>
> see the log of git

- git cat-file -p [hash]

> show the data of commit

- git commit -a [file]

> commit all the thing, = git add + git commit

- git checkout [hash]

> go back to the version
>
> `git checkout [branch]`
>
> > switch to another branch

- git diff

> show the different between commit

branch

- git brach -vv

> show the user

- git branch [name]

> create branch
>
> can use git log to see which node the brach is referred to

- git merge [branch]

> merge [branch] to the branch the head refer to
>
> `git merge -- continue` when the merge come cross the problem.

- git branch -d [branch]

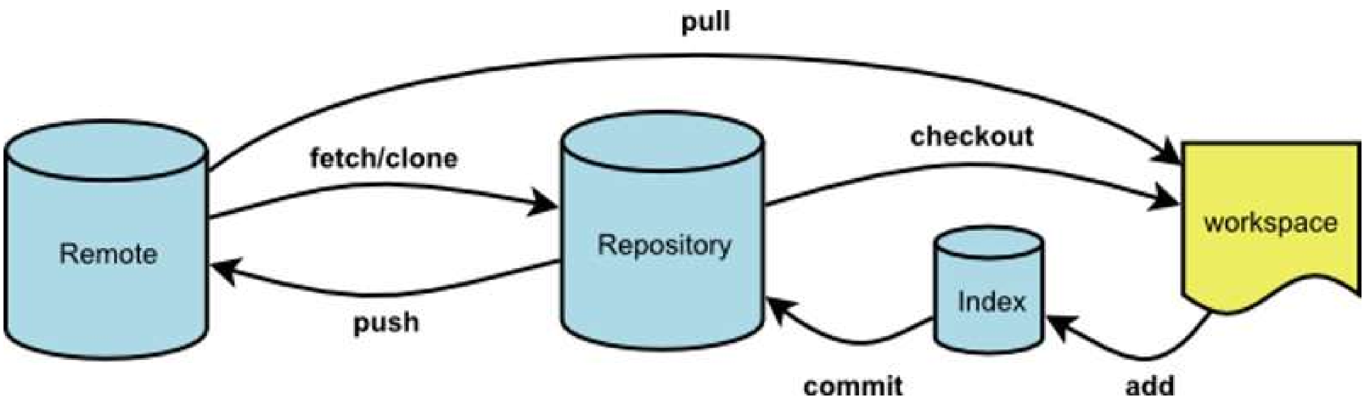
> delete branch

remote

- git remote add [name] [url]
- git push [remote] [local branch] [remote branch]

> you can see branch in `git log --all --graph --decorate`
>
> ex. git pushh origin master: master

- git clone



- git fetch

> get all the new thing

- git pull

> combine fetch and merge

- ~/.gitconfig

> setting

-

data

- journalctl

> log system

- less

> 分页

- sed

> a (add)
>
> c (reply)

> d (delete)
>
> s ()

- wc

> word count

正则表达式

> 菜鸟教程

- [ABC]

> 匹配所有含有[]内的字母
>
> 如[ab] 匹配[asdasdb]中aab三项

- [^ABC]

> 匹配所有不含ABC内的字母

- [A-Z]

> 所有大写字母

- .

> 除了换行符外所有字符，相当于[^\n\r]

- [\s]

> 所有空白符号

- [\S]

> 非空白字符

- \w

> 等于[A-Za-z0-9]

- $

> 匹配字符串结尾位置

- \

> 转义字符，将表达式中的语法字符转化成正常字符

- ()

> 标记一个子表达式的开始和结束位置。

- *

> 匹配前一个子表达式0或多次

- +

> 匹配前一个表达式1或多次

- ?

> 匹配前面的子表达式0或1次

- []

> 表示一个字符

basic operation

- cat /etc/issue

> see the system

- df

> see the disk in the system

- python

> run the sh in the desktop/个人/sh/python.sh

- `ln -s [A] [B]`
- `yum install lrzsz`
- `scp [local file] root@[ip]:[route file]`

---

1. 在~/ssh下创建authorized_keys `touch ~/.ssh/authorized_keys`
2. 复制自己的公钥并粘贴 `vim ~/ssh/authorized_keys`
3. 赋予权限 `chmod 700 ~/.ssh`
4. `chmod 700 ~/.ssh/authorized_keys`
