

Linux

2020年5月12日 21:02

简单的文件操作

打开终端操作`ctrl+alt+t`

home里存的用户 `cd noilinux/`

使用 `tree` 命令，可以列出一个文件夹下的所有子文件夹和文件（以树形结构来进行列出）

进入根目录`cd /`再输入`tree` 执行指令后，终端会像走马灯一样，遍历出系统里的所有目录和文件，这时按下 `ctrl + c` 键即可停止

(1) `pwd` 显示当前文件路径

(2) `ls` 显示当前文件夹下的所有文件 `-l`显示每个文件的详细信息

`ls -a`显示所有文件

`rw`可读可写可执行，三组分别对于根用户组内用户

(3) `cd` 切换目录

绝对路径，以`/`开头

相对于当前文件的相对路径，直接以某个文件夹开头

相对于每个用户家目录的路径`~/`

`cd ..` 表示返回上层目录

`cd .` 表示当前目录

`cd -` 上一次所在的目录

`cd ~` 表示回到当前用户的主目录，类似 Windows 的「回到桌面」

`cd /` 表示进入根目录，它是一切目录的父目录

(4) `mv` 移动

`mv a b` 将文件a移动到文件b内部

(1) b是一个文件夹，那么会将a移动到b内部

(2) b是一个文件，那么会把b覆盖掉(如果b存在，会覆盖掉；如果b不存在，相当于把a重命名成b)

(5) `cp` 复制

`cp a b` 将文件a复制到文件b或者说复制文件到指定目录下

(1) b是一个文件夹，那么会将a复制到b内部

(2) b是一个文件，那么会把b覆盖掉(如果b存在，会覆盖掉；如果b不存在，相当于把a重命名成b)

如果a是文件夹，那么需要加上`-r`指令

`cp test1.cpp ../`复制到上一层

(6) `rm` 删除

`rm a` 删除一个文件

`rm a -r` 删除一个文件夹

(7) `mkdir` 创建文件夹或者说创建目录

一次性创建多级目录 `mkdir -p one/two/three`

`tree one` 查看

(8) `cat` 看文件具体内容

(9) `g++ xxx.cpp -o xxx ./xxx` 编译加运行 `ctrl+c` 结束程序

使用 `touch` 命令可以新建文件

如果忘记了目录名、文件名或命令，可使用 `Tab` 键自动补全，还可避免输入错误；连续

按两次 `Tab` 可以显示全部候选结果。方向键上看之前命令 `history` 历史命令

`Ctrl + shift + c` `ctrl + shift + v`

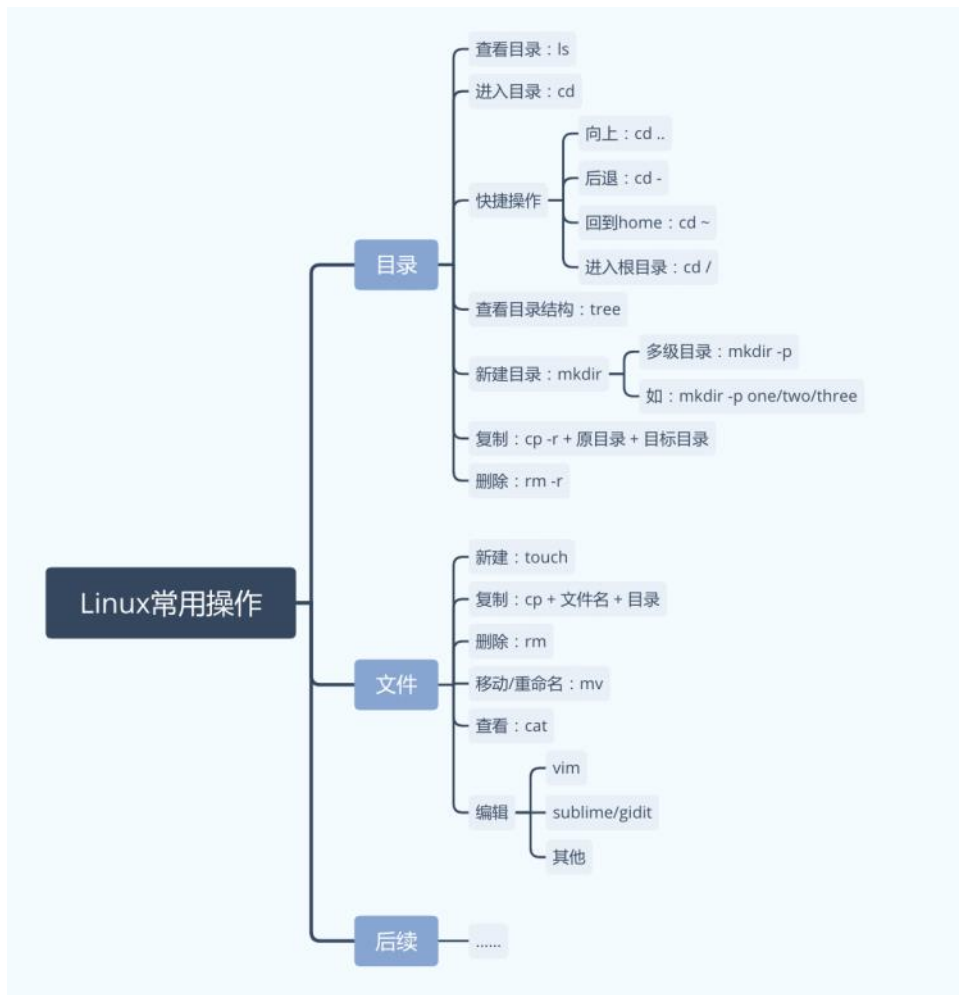
其他一些常用快捷键

按键	作用
<code>Ctrl+d</code>	键盘输入结束或退出终端
<code>Ctrl+s</code>	暂停当前程序，暂停后按下任意键恢复运行
<code>Ctrl+z</code>	将当前程序放到后台运行，恢复到前台为命令 <code>fg</code>
<code>Ctrl+a</code>	将光标移至输入行头，相当于 <code>Home</code> 键
<code>Ctrl+e</code>	将光标移至输入行末，相当于 <code>End</code> 键
<code>Ctrl+k</code>	删除从光标所在位置到行末
<code>Alt+Backspace</code>	向前删除一个单词
<code>Shift+PgUp</code>	将终端显示向上滚动
<code>Shift+PgDn</code>	将终端显示向下滚动

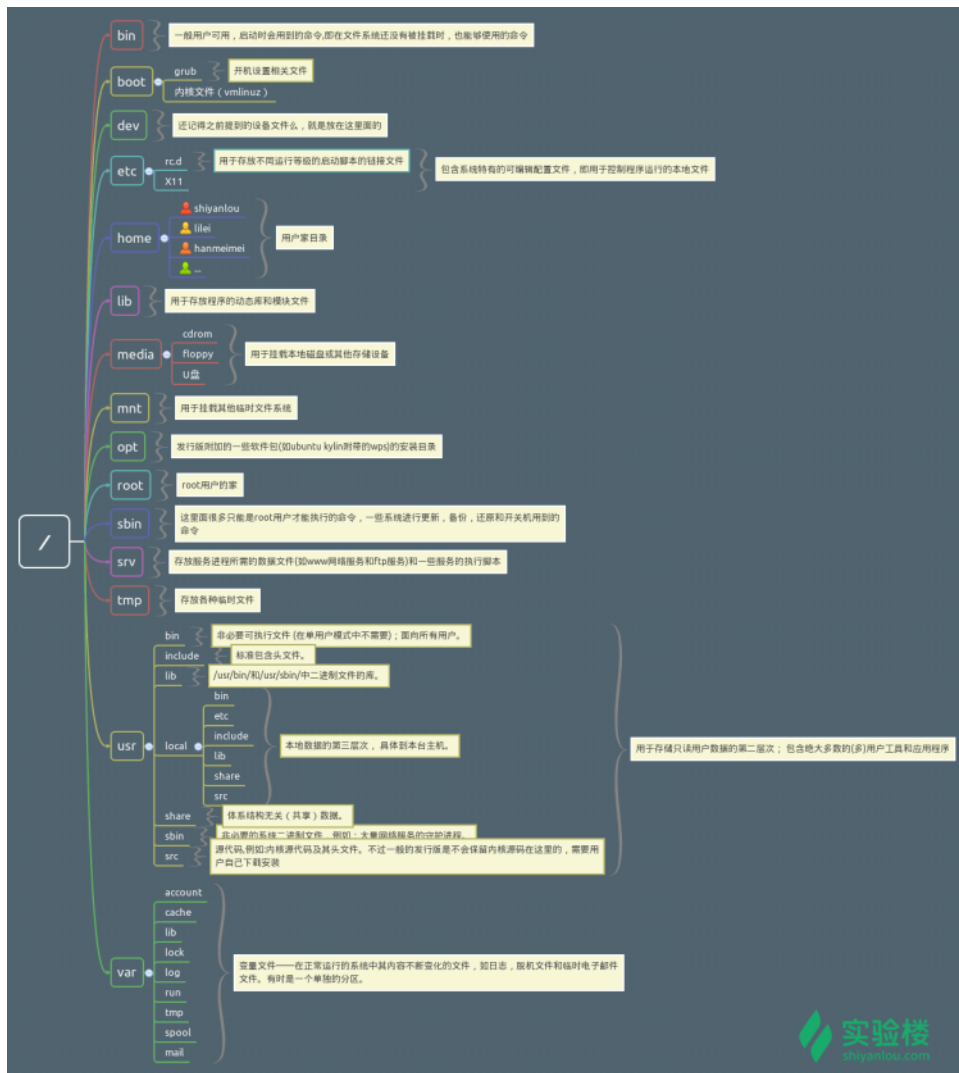
Shell 常用通配符：

字符	含义
<code>*</code>	匹配 0 或多个字符
<code>?</code>	匹配任意一个字符
<code>[list]</code>	匹配 list 中的任意单一字符
<code>[^list]</code>	匹配 除 list 中的任意单一字符以外的字符
<code>[c1-c2]</code>	匹配 c1-c2 中的任意单一字符 如： <code>[0-9][a-z]</code>
<code>{string1,string2,...}</code>	匹配 string1 或 string2 (或更多)其一字符串
<code>{c1..c2}</code>	匹配 c1-c2 中全部字符 如 <code>{1..10}</code>

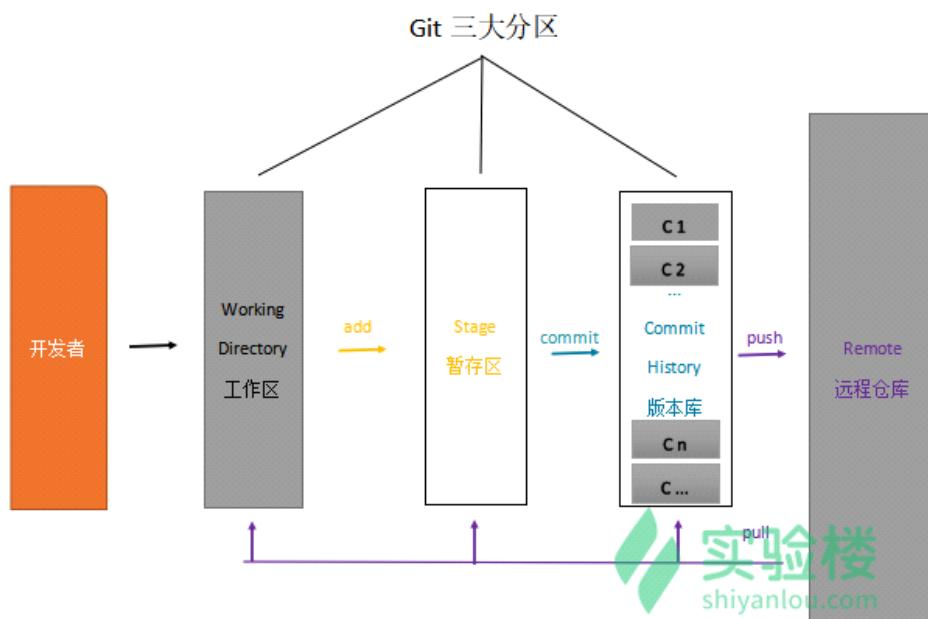




下图为 Linux 操作系统目录结构的简介，最左边就是根目录：



如果对多个文件或目录进行了增删改, 可以使用 `git add .` 命令全部添加到暂存区



注意这里有个概念, 当我们修改了工作区, `git add` 命令是把这些修改添加到暂存区, 暂存区记录的只是修改。如果要撤销暂存区的修改怎么办? 根据上图的提示, 执行 `git reset -- [文件名]` 或者 `git rm --cached [文件名]` 命令即可


如果省略最后的文件名，把命令写成 `git reset --` 即可把暂存区的全部修改撤销。好，现在暂存区的修改被撤销，又回到了工作区

关于查看提交历史记录的命令，有些常用的选项介绍一下：

`git log [分支名]` 查看某分支的提交历史，不写分支名查看当前所在分支
`git log --oneline` 一行显示提交历史
`git log -n` 其中 `n` 是数字，查看最近 `n` 个提交
`git log --author [贡献者名字]` 查看指定贡献者的提交记录
`git log --graph` 图示法显示提交历史

现在介绍一个超级实用、使用频率极高但几乎所有 Git 教程都不重视的命令 `git branch -avv`，它用来查看全部分支信息：

```
* master          5c041ad [origin/master: 领先 1] commit file one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 2b96af0 Initial commit
(END)
```



上图有三行信息，依次说明：

第一行，开头的星号表示当前所在分支，绿色的 `master` 是分支名，之所以是绿色，也是因为它是当前所在分支。后面第二项是版本号，第三项中括号里面蓝色的字，表示此分支跟踪的远程分支的名字，当然啦，这也是克隆远程仓库到本地时的默认设置 -- 创建 `master` 分支并自动跟踪远程同名分支；冒号后面黑色文字表示本地分支领先其跟踪的远程分支一个提交。最后一项是提交时填写的备注信息。

第二行，是 Git 指针信息，它指向远程仓库的 `master` 分支，这行信息暂不重要。

第三行，远程分支信息，详见第一行的解释

首先执行 `git reset --soft HEAD^` 撤销最近的一次提交，将修改还原到暂存区。`--soft` 表示软退回，对应的还有 `--hard` 硬退回，后面会讲到，`HEAD^` 表示撤销一次提交，`HEAD^^` 表示撤销两次提交，撤销 `n` 次可以简写为 `HEAD~n`

`git add .` `git commit` `git push -f`

本地仓库 `commit` 变化记录

有 `git reflog` 命令，它会记录本地仓库所有分支的每一次版本变化。

执行 `git reset --hard HEAD@{2}` 命令，回退到某个版本，`git push -f`

rm -rf test 删除原仓库

使用 SSH 的好处主要有两点：

- 免密码推送，执行 git push 时不再需要输入用户名和密码了；
- 提高数据传输速度

执行 git branch [分支名] 可以创建新的分支

此命令创建新分支后并未切换到新分支，还是在 master 分支上，执行 git checkout [分支名] 切换分支，checkout 也是常用命令，先给它设置别名，然后切换分支

```
shiyanolou:shiyanolou/ (master) $ git config --global alias.ch checkout
shiyanolou:shiyanolou/ (master) $ cat -n ~/.gitconfig
 1 [user]
 2     email = 1195581533@qq.com
 3     name = Manchangdx
 4 [alias]
 5     st = status
 6     br = branch -avv
 7     com = commit -m
 8     ch = checkout
shiyanolou:shiyanolou/ (master) $ git ch dev
切换到分支 'dev'
shiyanolou:shiyanolou/ (dev) $
```



创建新分支还要手动切换太麻烦，介绍另一个常用的命令 git checkout -b [分支名] 创建分支并切换到新分支

执行 git push [主机名] [本地分支名]:[远程分支名] 即可将本地分支推送到远程仓库的分支中，通常冒号前后的分支名是相同的，如果是相同的，可以省略:[远程分支名]，如果远程分支不存在，会自动创建。命令可以简写为 git push origin dev1

执行这个命令 git branch -u [主机名/远程分支名] [本地分支名] 将本地分支与远程分支关联，或者说使本地分支跟踪远程分支，如果是设置当前所在分支跟踪远程分支，最后一个参数本地分支名可以省略不写。这个命令的 -u 选项是 --set-upstream 的缩写

执行 git branch --unset-upstream [分支名] 即可撤销该分支对远程分支的跟踪，同样地，如果撤销当前所在的分支的跟踪，分支名可以省略不写

在推送时就自动跟踪远程分支呢？有的，在推送的时候，加个 --set-upstream 或其简写 -u 选项即可，现在切换到 dev 分支试一下这个命令

```
git push -u origin dev
```

删除远程分支，使用 git push [主机名] :[远程分支名]，如果一次性删除多个，可以这样：git push [主机名] :[远程分支名] :[远程分支名] :[远程分支名]。此命令的原理是将空分支推送到远程分支，结果自然就是远程分支被删除。另一个删除远程分支的命令：git push [主机名] --delete [远程分支名]。删除远程分支的命令可以在任意本地分支中执行

```
git push origin :dev
git push origin --delete dev1
```

给本地分支改名 git branch -m [原分支名] [新分支名]，若修改当前所在分支的名字，原分支名

可以省略不写

```
git branch -m ved
```

当前所在的分支不能被删除。切换到 master 分支，然后执行 `git branch -D ved dev1` 命令

```
git ch master
```

```
git branch -D ved dev1
```

Fork：在别人的仓库中点此按钮会克隆一个完全一样的仓库到你自己的账号中，包括所有分支、提交等，但不会克隆 issue（本节后面会讲到），当此仓库发生版本变化，不会自动同步到你克隆

的仓库里，反之亦然。

ssh关联授权

```
git clone ...
```

```
cd test
```

```
ssh-keygen
```

将 `~/.ssh/id_rsa.pub` 文件中的公钥内容复制出来，然后在 GitHub 网页上添加公钥，Title 自定义，把剪切板中的内容粘贴到 Key 中，点击绿色按钮添加 SSH Key 即可

回到仓库主目录，点击下图所示的绿色按钮，点击紫色框中的 “Use SSH”，然后复制这个链接

在实验环境里删除原仓库，使用此链接重新克隆仓库。克隆仓库是需要确认连接，输入 yes 即可

```
shiyanolou:shiyanolou/ (master) $ cd
shiyanolou:~/ $ rm -rf shiyanolou
shiyanolou:~/ $ git clone git@github.com:Manchangdx/shiyanolou
fatal: 仓库 'git@github.com:Manchangdx/shiyanolou' 不存在
shiyanolou:~/ $ git clone git@github.com:Manchangdx/shiyanolou
正克隆到 'shiyanolou'...
The authenticity of host 'github.com (13.229.188.59)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,13.229.188.59' (RSA) to the list of known hosts.
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
接收对象中: 100% (7/7), 完成.
shiyanolou:~/ $
```

