学习报告

汤宁

2015 年 10 月

Contents

1. Linux 基础学习

1.1 对 Linux 的一点小理解

1.1.1 目录树结构

Linux 中的所有数据都是以文件形态呈现的，以根目录为主，呈现分支状的目录结构。

1.1.2 “挂载”—— 文件系统与目录树的关系

进入一个目录读取该分区，进入的目录称为“挂载点”。

1.2 Ubuntu 安装分区理解

/：根目录，在 linux 中所有文件和目录都是从根目录开始的。

/swap：交换空间，当内存不足时，把一部分磁盘空间虚拟成内存使用从而解决内存不足的问题。

/home：用户目录，存放普通用户数据。

/opt：第三方软件安装目录（应用程序软件包）。

/boot(单系统时不用安装)：存放系统启动相关程序（开机与内核文件）。

1.3 Linux 基本终端指令学习

1.3.1 查看文件与目录—— ls

ls [options]

1）当前目录下所有文件和文件夹。

2）-a：当前目录下所有文件（包括可见文件和隐藏文件）。

3）-l：当前目录下所有可见文件详细属性。

4）-al：当前目录下所有文件的详细属性。

1.3.2 Linux 文件权限相关指令

改变所属用户组 ——chgrp

chgrp [-R] 账号名称文件或目录

-R：递归更改，更改目录下的所有文件和目录。

改变文件所有者 ——chown

1.chown [-R] 账号名称文件或目录

-R：递归更改，更改目录下的所有文件和目录。

2.chown root:root 文件名称

将该文件的所有者和用户组改回为 root。

改变权限 ——chmod

基本权限的作用：

一. 对于目录：

r: 当对目录具有权限时，可以查询该目录下文件名

w: 可以更改目录结构列表

1. 新建新的文件与目录

2. 删除已存在的文件和目录

3. 将已存在的文件和目录重命名

4. 转移该目录内文件和目录位置

x: 用户能否进入该目录成为工作目录

二. 对于文件：

r: 读取文件实际内容

w: 编辑或新增修改文件内容，但不包含删除文件

x: 可以被系统执行

PS: 对于存放在某用户文件夹下的文件和目录，该用户在此文件或目录下具有 rwx 的完整权限

1. 数字类型改变权限：

chmod [-R] xyz(权限属性，分别代表所有者，用户组和其他人的权限）文件或目录

-R：递归更改，更改目录下的所有文件和目录。

r=4,w=2,x=1

2. 符号类型改变权限：

u: 所有者 g: 用户组 o: 其他人 a: 所有

chmod a+x 文件名

chmod a-x 文件名

chmod u=rwx,go=rx 文件名 ...

修改默认权限—— umask

umask——查看默认权限（默认值需要减掉的权限）

umask -S 查看具体默认权限

默认权限属性：

1. 用户创建“文件”，默认没有可执行（x）权限，最大为 666

-rw-rw-rw-

2. 用户新建“目录”，由于 x 与是否可以进入此目录有关，默认所有权限开放为 777

drwxrwxrwx

设置文件隐藏属性——chattr

chattr [+-=][options] 文件或目录名

i: 能使文件不能被删除，改名，设置连接，写入或添加数据，只有 root 能设置此属性。

a: 设置后，文件只能增加数据，不能删除也不能修改，只有 root 能设置此属性。

显示文件隐藏属性——lsattr

1.3.3 切换目录—— cd

cd [相对路径或绝对路径]

“.” 和 “..” 分别代表此层目录和上层目录。

1）cd ～：回到自己的主文件夹。

2）cd ..：回到目前所在目录的上层目录。

3）cd -：回到刚才的目录。

4）cd ../目标目录：相对路径的写法，可以由当前目录到达与其属于同一上层目录的目标目录。

1.3.4 复制，删除与移动—— cp,rm,mv

1.cp [options] 源文件 目标文件

1）-r：递归复制，可用来复制目录。

2）-p：连同文件属性一齐复制

2.rm [options] 文件或目录

-r：递归删除，常用于目录删除。

3.mv 源文件 目标文件

可以用来改名。

1.3.5 新建目录与删除空目录—— mkdir,rmdir

mkdir [options] 目录名称

-p：可创建多层目录。

rmdir [options] 空目录名称

-p：递归删除一系列空目录。

1.3.6 文件名的查找—— find

find / -name 文件名

find / -name xxx* 查找以 xxx 开头的文件

find / 目录 -name '*xxx*' 查找目录下文件名包含 xxx 的文件

find / -mtime+4

PS: 除了 find 命令外，还可以通过 whereis 和 locate 来查找，但 whereis 和 find 是从数据库中进行查找的，虽然查找速度较快但不一定能找到我们需要找的文件和目录，find 虽然费时，但是从硬盘中直接查找。

mtime: 内容更改时间
ctime: 状态更改时间（权限和属性）
atime: 文件内容被取用的时间
+n:n 天前（不含 n 天本身）
-n:n 天内（含 n 天本身）
n:n 天之前的“一天之内”

1.3.7 新建空文件—— touch

touch 文件名

新建了空文件，可以用 'vim 文件名' 打开该空文件并进入编辑模式添加数据。

1.4 vim 程序编辑器

vim 文件名

可进入一般模式，按 'i' 进入编辑模式，按 'Esc' 退回一般模式。

vim 一般模式的基本按键功能：

Ctrl+[f]——屏幕向下移动一页

Ctrl+[b]——屏幕向上移动一页

0——移动到这一行最前面的字符处

$ ——移动到这一行最后面的字符处

G——移动到文件最后一行（nG 移动到文件第 n 行）

gg——移动到文件第一行 (相当于 1G)

n+[Enter]——n 为数字，光标向下移动 n 行

x,X——x 为向后删除一个字符，X 为向前删除一个字符

dd——删除光标所在那一行（ndd 删除光标所在向下 n 行）

yy——复制光标所在那一行（nyy 复制光标所在向下 n 行）

p,P——p 为将已复制的数据在光标下一行粘贴，P 为粘贴在光标上一行

u——复原前一个操作

Ctrl+[r]——重做上一个操作

:q——离开 vim 一般模式

:q!——强制离开不保存文件

:wq——保存后离开

1.5 学习 Linux 的心得体会

在还没有接触 Linux 之前，我就已经听说过他的大名，但那个时候只是知道学习 Linux 有利于找工作罢了, 而通过这段时间对 Linux 终端指令的学习，使我开始逐渐意识到 Linux 的强大之处。首先，Linux 系统上的文件都是可以通过终端来进行更改设置的。之前我安装了 Codeblocks 和 Opencv 之后，对两者的全局编译链接进行了设置，设置后发现虽然每次关闭 Codeblocks 前都有保存过，但是下次打开后之前设置的都不见了，后来在老师的帮助下才发现 Codeblocks 安装包的所有者和用户组都是 root，也就是说以我本人的名义是不能对 Codeblocks 进行设置更改的，于是我更改了安装包的所有者和用户组，问题才得以解决。通过这个小问题，使我对 Linux 的态度产生了极大的转变，我觉得 Linux 系统上文件都可以更改这一点就能很好的体现出它的开源性，虽然这样也会增大一不小心改动系统文件导致系统发生故障的风险。如果单单通过图形界面来操作 Linux 的话虽然有的时候也许会很方便，但是遇到比如权限一类的问题，图形界面是行不通的，很显然在这种情况下，终端指令无疑是最好的选择。此外，Linux 的保护功能也很强大，我的电脑安装了 Win7 和 Ubuntu 双系统后，在 Win7 界面下只能看到它自己的磁盘分区而在 Ubuntu 界面下不仅可以看到它自己的还可以看到 win7 的磁盘分区，不同的是在 Ubuntu 下所有数据都是以文件形式存在的。Linux 独特的目录树形式和挂载来读取分区也是比较有意思的地方。总之，这段时间对 Linux 的学习虽然还不是那么深入，但也算

是收获颇丰，同时我还明白了一个道理：要想学好 Linux 还是得通过不断的实践才能更好的使用它，另外思考每个命令代表的意义也是很重要的。

2. 学习 LaTeX 的心得体会

LaTeX 是我最后安装的软件，我一直都觉得打印数学公式是一件很头疼的事情，最早的时候是从用 word 开始的,word 中虽然带有大量数学公式可以编辑但是也有一部分不能修改，后来又知道有款专门打出数学公式的软件叫 mathtype 可以和 word 很好的结合，然而此款软件却需要付费。所以学习 之于我最主要的地方就是它能打出漂亮工整的数学公式, 而如何使用 LaTeX 来画出所需要的图表又是个大问题，在 LaTeX 中比较简单的图表还是很容易可以得到的，而一旦遇到不规则表格，LaTeX 繁琐的代码就让我痛不欲生了。但是很显然它又有一个优势就是图形种类繁多，只有不会实现的没有实现不出来的。同时 LaTeX 强大的排版方式也是 word 望尘莫及的，使用 word 对长篇文章进行排版时很多文本参数需要不断的手工调整，而相反使用 LaTeX 排版时各种细节都可以统一规划设置，比较格式化。此外，LaTeX 与 word 相比显然排版更加正式，更适合论文和书籍的发表。正如老师所说的 word 是“所见即所得”，而 LaTeX 则是“所想即所得”。总之，LaTeX 的功能还是很强大的，对于撰写学术论文无疑是最好的选择。

3. Github 的学习和理解

3.1 对 Github 工作流程的理解

1. 用户 A 提交项目至本地仓库再提交到远程仓库；
2. 用户 B 从远程仓库克隆项目修改后再提交；
3. 用户 A 从远程仓库获取新的数据后再和本地仓库已有数据进行合并，然后提交；
4. 各个用户不断重复 2 和 3 的过程并最终完成项目；

总之，几个人合作使用 Github 是一个更新本地仓库数据，合并，提交的循环的过程。

3.2 分布式版本控制系统——Github

分布式版本控制系统：客户端不仅提取最新版本文件快照，而是把原始的代码仓库完整的镜像下来，协同工作的任何一处服务器发生故障可以用任何一个镜像出来的本地仓库恢复，每一次提交都是对代码仓库的完整备份。

Git 的特别之处：只关心文件数据整体是否发生变化而大多数系统只关心文件内容具体差异，它不保存前后变化差异数据，而是把变化的文件作快照后，记录在一个微型文件系统，每次提交更新纵览文件指纹信息 (在保存到 Git 前，所有数据进行检验和计算并将此结果作为数据唯一标识和索引，通过对文件内容或目录结构计算出一个哈希值作为指纹字符串，Git 的工作完全依赖于哈希值，所有保存在 Git 数据库中的东西都用此哈希值来索引而不是靠文件名）作一快照，然后保存一个指向快照的索引，若文件没有变化时不再保存，只对上次保存的快照作一连接。

3.2.1 Git 的三种文件状态和三个工作区域

- 三种文件状态：已修改，已暂存和已提交
  1. 已修改：已修改，但还未提交保存
  2. 已暂存：把已修改文件放入下次提交时要保存的清单中
  3. 已提交：已被安全地保存在本地数据库中

- 三种工作区域：Git 本地数据目录，工作目录及暂存区域（实质是个文件，一般放在 Git 目录中）

3.2.2 Git 基本工作流程

1. 在工作目录中修改文件
2. 对修改的文件做快照并保存到暂存区域
3. 提交更新, 将保存在暂存区域的文件快照转储到 Git 目录中

3.2.3 初次运行 Git 前的配置

一般在新的系统上要先配置自己的 Git 工作环境，配置工作只需一次，以后升级还会沿用现在的配置
git config 专门用来配置或读取相应的工作环境变量
可以放在三个地方：
/etc/gitconfig——系统中对所有用户普遍适用的配置
~/.gitconfig——用户目录下的配置文件只适用于该用户
.git/config——当前项目中配置文件仅对当前项目有效

3.2.4 取得 Git 项目仓库的方法

1. 在现存目录下，通过导入所有文件来创建新的 Git 仓库到此项目所在目录。
git init 命令生成.git 隐藏仓库

2. 从已有 Git 仓库克隆出一个新的镜像仓库，收取项目历史中的所有数据“每个文件的每个版本”，服务器上的数据克隆后本地也有。

git clone git@github.com: 用户名/仓库名.git——克隆用户的远程仓库到本地形成本地仓库

PS: 初次克隆某个仓库时，工作目录中所有文件属于已跟踪状态，状态为未修改。

3.2.5 Git 分支

在 Git 提交时，会保存一个提交对象，它包含一个指向暂存文件快照的指针，作者和相关所属信息以及一定数量（可能没有）指向该提交对象直接祖先的指针，第一次提交是没有直接祖先的, 每个提交对象包含着指向树对象 (有点像目录) 的指针，而每个树对象又管理着一些子树对象和以 blob 类型存储的文件快照，必要时可以重现快照内容。修改后再次提交，这次的提交对象会包含一个指向上次提交对象的指针。

- Git 分支：本质上是指向提交对象的可变指针，master 为默认名字，每次提交时后自动前移。
- HEAD 指针：指向正在工作的本地分支。

git branch：在当前提交对象上新建分支；

git checkout: 可以移动 HEAD 指针从而切换分支;

git checkout -b: 新建并切换到分支;

git branch -d: 删除分支。

PS: 一次克隆后会建立本地分支 master 和远程分支 origin/master. 他们都指向 origin/master 分支的最后一次提交，只要不和服务器通讯，origin/master 指针不会移动，要用 git fetch 来进行同步更新本地数据库。

3.2.6 merge(合并) 和 rebase(衍合) 的区别

- git merge（合并）：Git 利用两分支末端和他们的共同祖先进行三方合并，并且创建一个指向这两个分支末端的提交对象。另外，如果顺着一个分支走下去可以到达另一个分支，那么 Git 在合并两者时只需把指针前移。
- git rebase(衍合)：回到两个分支（所在分支和要衍合的分支）的共同祖先，提取每次提交产生的差异，把差异保存到临时文件中，再转到要衍合的分支并依序施用差异补丁文件。

衍合后的内容和合并后的内容一样，只是历史记录不用，衍合的历史记录更整洁，因为其抛弃了一些现存的提交对象并创造了一些新的提交对象。

3.2.7 fetch 和 pull 的区别

Git 中从远程的分支获取最新版本到本地有 2 个命令：

- git fetch：相当于是从远程获取最新版本到本地，不会自动合并
- git pull：相当于是从远程获取最新版本并合并到本地

3.3 学习 Github 的心得体会

通过这段时间对 Github 的使用和学习，我觉得 Github 更适合团队合作项目的开发，当然对于个人更新修改一些比较重要的东西也是个好方法，因为每次提交都是对数据的备份，所以不用担心数据会丢失。此外由于 Github 是一款开源的版本控制系统，因此大量的代码数据都可以供人们共享使用，Github 不失为一个获取代码的绝佳资源库。学习 Github 的使用方法是件很有趣的事，因为可能一不小心就会发生各种问题，这些问题都需要我们去积累尝试并解决。