## **Linux中常用到的命令**

　　显示文件目录命令ls        如ls

　　切换目录命令cd        如cd /home

　　建立子目录mkdir           如mkdir xiong

　　删除子目录命令rmdir       如rmdir /mnt/cdrom

　　删除文件命令rm            如rm /ucdos.bat

　　文件复制命令cp            如cp /ucdos /fox

　　获取帮助信息命令man      如man ls

　　显示文件的内容less        如less mwm.lx

　　重定向与管道type          如type readme>>direct，将文件readme的内容追加到文direct中

## **Linux文件属性有哪些？（共十位）**

　　-rw-r--r--那个是权限符号，总共是- --- --- ---这几个位。

　　第一个短横处是文件类型识别符：-表示普通文件；c表示字符设备（character）；b表示块设备（block）；d表示目录（directory）；l表示链接文件（link）；后面第一个三个连续的短横是文件所有者权限位（User），第二个三个连续短横是组权限位（Group），第三个三个连续短横是其他用户权限位（Other）。每个权限位有三个权限，r（读权限），w（写权限），x（执行权限）。如果每个权限位都有权限存在，那么满权限的情况就是：-rwxrwxrwx；权限为空的情况就是- --- --- ---。

Linux中的权限有r(读) w(写) x(执行)，分别用数字4,2,1代表。

Umask是设置系统创建文件时的默认权限，是创建文件权限补码，对文件来说最大值是6，文件默认666，目录默认777，减去umask就是结果。

Umask设为为244，则创建的文件默认权限是422，文件的第一位是‘-’也就是-r---w--w-

　　权限的设定可以用chmod命令，其格式位：chmod ugoa+/-/=rwx filename/directory。例如：

　　一个文件aaa具有完全空的权限- --- --- ---。

　　chmod u+rw aaa（给用户权限位设置读写权限，其权限表示为：- rw- --- ---）

　　chmod g+r aaa（给组设置权限为可读，其权限表示为：- --- r-- ---）

　　chmod ugo+rw aaa（给用户，组，其它用户或组设置权限为读写，权限表示为：- rw- rw- rw-）

　　如果aaa具有满权限- rwx rwx rwx。

　　chmod u-x aaa（去掉用户可执行权限，权限表示为：- rw- rwx rwx）

　　如果要给aaa赋予制定权限- rwx r-x r-x，命令为：

　　chmod u=rwx，go=rx aaa

## ext2、ext3与ext4的区别

① ext2是非日志文件系统，ext3是日志文件系统，而ext4修改了ext3中部分重要的数据结构

②

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Block大小 | 1KB | 2KB | 4KB | 4KB(EXT4) |
| 最大单一文件 | 16GB | 256GB | 2TB | 16TB |
| 最大文件系统总容量 | 2TB | 8TB | 16TB | 1EB |

③ 无限数量的子目录。 Ext3 目前只支持 32,000 个子目录，而 Ext4 支持无限数量的子目录。

## G++

1. （程序 g++ 是将 gcc 默认语言设为 C++ 的一个特殊的版本，链接时它自动使用 C++ 标准库而不用 C 标准库。）

2. 选项

**预处理-E**： 使 g++ 将源代码用编译预处理器处理后不再执行其他动作。下面的命令预处理源码文件 helloworld.cpp 并将结果显示在标准输出中，预处理过的文件的 GCC 后缀为 .i，它可以通过 -o 选项来生成，例如： $ gcc -E helloworld.cpp -o helloworld.i

**编译-S**： 指示编译器将程序编译成汇编语言，输出汇编语言代码而后结束。下面的命令将由 C++ 源码文件生成汇编语言文件 helloworld.s： $ g++ -S helloworld.cpp

**汇编-c**： 用来告诉编译器编译源代码但不要执行链接，输出结果为对象文件。文件默认名与源码文件名相同，只是将其后缀变为 .o。

**链接-o** file ：   生成输出文件file，可以生成任何中间或结果文件，可以和前三个参数结合使用，如果单独使用则生成最终的可执行文件。

-g： 在可执行程序中包含标准调试信息

## gdb

1. 编译时需要加-g参数，如：g++ test.c -o test -g;
2. 加载文件：file filename
3. 运行：r
4. 设置断点：b linenum、b funname如：b 6或 b main
5. 下一步：s
6. 查看某个变量值：p valname
7. 执行到下一个断点：c
8. 删除所有断点：d
9. 查看汇编代码，进行汇编级跟踪：display /i $pc
10. 汇编级查看下一步：si
11. b \*funname:在函数funname的prolog代码处设置断点（prolog、epilog，分别表示编译器在每个函数的开头和结尾自行插入的代码）
12. 查看寄存器中的值： i r
13. 退出：q

## Git

1. 设置ssh

ssh-keygen -t rsa -C “your email”

一直回车即可

在c:\用户\当前用户文件夹\.ssh\id\_rsa.pub文件，将其用记事本打开，复制其中的内容到Github账户下添加该SSH Key。

测试：ssh -T [git@github.com](mailto:git@github.com)

输入”yes”出现”Hi name……”即为成功。

在Github上创建Repository，并记下类似[git@github.com：sucunbaby/mytest.git](mailto:git@github.com：sucunbaby/mytest.git)的地址。在创建远程仓库时，这就是远程仓库的url。

1. 安装成功后，

git config --global user.name “name”

git config --global user.email “email”

1. 创建版本库：

切换到目标目录：cd F：

创建一个目录：mkdir 目录名

切换到该目录：cd 目录名

显示当前路径：pwd

设置为仓库：git init（目录下多一个.git的隐藏文件）

将文件添加到目录库：git add 文件名 -m “注解”

查看是否有文件未提交：git status

查看某个文件到底改了什么内容：git diff 文件名

提交修改：git commit

创建分支：git branch branchname(只有提交过一次才可以创建分支)

删除分支：git branch -d branchname

合并分支：git merge branchname

切换分支：git checkout branchname

查看历史版本：git log

退回：

* 1. git reset –hard HEAD^（退回上个版本，同样退回上上个版本：git reset –hard HEAD^^）
  2. git reset –hard HEAD~100（退回前100个版本）

查看版本号：git reflog

回到某个版本：git reflog 版本号

查看远程仓库：git remote

创建远程仓库：git remote add name url(如：[git@github.com:sucunbaby/mytest.git](mailto:git@github.com:sucunbaby/mytest.git))

从远程仓库抓取：git fetch remotename branchname(**:temp**-创建一个临时分支)

合并:git merge temp (--allow-unrelated-histories:允许不相干的文件合并)

推送数据到远程仓库：git push remotename branchname

查看远程仓库信息：git remote show origin

远程仓库重命名：git remote rename remotename newname

远程仓库删除：git remote rm remotename

例如：

git fetch origin master:temp(origin时远程仓库的名字，master时默认分支，temp是临时创建的分支)

git diff temp（比较temp分支和当前分支的区别）

git merge temp（将temp分支合并到当前分支）

git branch -d temp（删除temp分支）