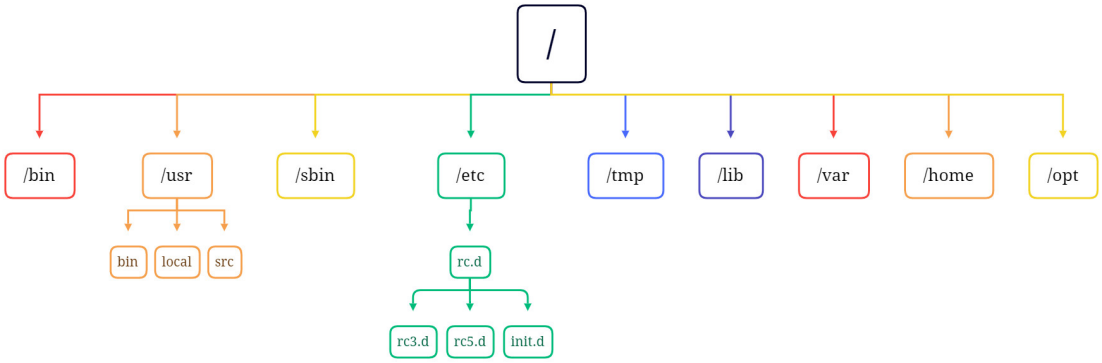
Linux

Linux 目录介绍



- bin 存放二进制可执行文件
- boot 存放系统引导时使用的各种文件
- dev 存放设备文件
- etc 存放系统配置文件
- home 存放系统用户的文件
- lib 存放程序运行所需的共享库和内核模块
- opt 额外安装的可选应用程序包所放置的位置
- root 超级用户目录
- sbin 存放二进制可执行文件，只有 root 用户才能访问
- tmp 存放临时文件
- usr 存放系统应用程序
- var 存放运行时需要改变数据的文件，例如日志文件

Linux 常用命令

命令	对应英文	作用
ls	list	查看当前目录下的内容
pwd	print work directory	查看当前所在目录
cd [目录名]	change directory	切换目录
touch [文件名]	touch	如果文件不存在，新建文件
mkdir [目录名]	make directory	创建目录
rm [文件名]	remove	删除指定文件

> ⚠注意：
> 在执行 Linux 命令时，提示信息如果显示为乱码，这是由于编码问题导致，只需要修改 Linux 的编码即可，命令如下：

```
echo 'LANG="en_US.UTF-8"' >> /etc/profile #把LANG="en_US.UTF-8"这句话添加到etc下的
source /etc/profile #重新加载etc下的profile文件
```

• Linux 命令使用技巧

- Tab键自动补全
- 连续两次Tab键，给出操作提示
- 使用上下箭头快速调出曾经使用过的命令
- 使用 `clear` 命令或者Ctrl+L快捷键实现清屏

• Linux 命令格式

```
command [-options] [parameter]
```

说明：

- command：命令名
- [-options]：选项，可用来对命令进行控制，也可以省略
- [parameter]：传给命令的参数，可以是零个、一个或多个

> ⚠注意：

> [] 代表可选
>
> 命令名、选项、参数之间有空格进行分隔

• 文件目录操作命令

- 文件目录操作命令 ls

作用：显示指定目录下的内容

语法：

```
ls [-a/-l/-al] [dir]
```

说明：

- `-a` 显示所有文件及目录（`.` 开头的隐藏文件也会列出）
- `-l` 除文件名称外，同时将文件型态（`d` 表示目录，`-` 表示文件）、权限、拥有者、文件大小等信息详细列出
- 多个选项可以连写，如 `-a` 和 `-l` 可以连写为 `-al`

> “
>
> ⚠注意：
>
> 由于我们使用 `ls` 命令时经常需要加入 `-l` 选项，所以Linux为 `ls -l` 命令童工了一种简写方式，即 `ll`

- 文件目录操作命令 cd

作用：用于切换当前工作目录，即进入指定目录

语法：

```
cd [dirName]
```

特殊说明：

- `~` 表示当前用户的home目录
- `.` 表示目前所在的目录
- `..` 表示目前目录位置的上级目录

文件目录操作命令 cat

作用：用于显示文件的内容

语法：

```
cat [-n] fileName
```

说明：

- `-n` ： 有1开始对所有输出的行数编号

文件目录操作命令 more

作用：以分页的形式显示文件内容

语法：

```
more fileName
```

操作说明：

- 回车键(`enter`)--------向下滚动一行
- 空格键(`space`)--------向下滚动一屏
- `b`----------返回上一屏
- `q`或者`Ctrl+C` --------退出more

文件目录操作命令 tail

作用：查看文件末尾的内容

语法：

```
tail [-f] fileName
```

说明：

- `-f` ： 动态读取文件末尾内容并显示，通常用于日志文件内容的输出

举例：

```
tail /etc/profile      #显示/etc目录下的profile文件末尾10行的内容
tail -20 /etc/profile  #显示/etc目录下的profile文件末尾20行的内容
tail -f /itcast/my.log #动态读取/itcast目录下的my.log文件末尾内容并显示
```

文件目录操作命令 mkdir

作用：创建目录

语法：

```
mkdir [-p] dirName
```

说明：

- `-p` ：确保目录名称存在，不存在就创建一个。通过此选项，可以实现多层目录同时创建

举例：

```
mkdir itcast      #在当前目录下，建立一个名为itcast的子目录
mkdir itcast/test #在工作目录下的itcast目录中建立一个名为test发子目录，若itcast目录不存在，则创建一个
```

文件目录操作命令 rmdir

作用：删除空目录

语法：

```
rmdir [-p] dirName
```

说明：

- -p ： 当子目录被删除后使父目录为空目录的话，则一并删除

举例：

```
rmdir itcast         #删除名为itcast的空目录
rmdir -p itcast/test #删除itcast目录中名为test的子目录，若test目录删除后itcast目录变为空目录，则也被删除
rmdir itcast*        #删除名称以itcast开始的空目录
```

文件目录操作命令 rm

作用：删除文件或目录

语法：

```
rm [-r/-f/-rf] name
```

说明：

- `-r` ：将目录及目录中所有文件（目录）逐一删除，即递归删除
- `-f` ：无需确认，直接删除

• 拷贝移动命令 cp

作用：用于复制文件或目录

语法：

```
cp [-r] source dest
```

说明：

- `-r` ：如果复制的是目录需要使用此选项，此时将复制该目录下所有的子目录和文件

举例：

```
cp hello.txt itcast/      #将hello.txt复制到itcast目录中
cp hello.txt ./hi.txt     #将hello.txt复制到当前目录，并改名为hi.txt
cp -r itcast/ ./itheima/  #将itcast目录和目录下所有文件复制到itcast目录下
cp -r itcast/* ./itheima/ #将itcast目录下所有文件复制到itheima目录下
```

• 拷贝移动命令 mv

作用：为文件或目录改名、或将文件或目录移动到其他位置

语法：

```
mv source dest
```

举例：

```
mv hello.txt hi.txt          #将hello.txt改名为hi.txt
mv hi.txt itheima/           #将文件hi.txt移动到itheima目录中
mv hi.txt itheima/hello.txt #将hi.txt移动到itheima目录中，并改名为hello.txt
mv itcast/ itheima/          #如果itheima目录不存在，将itcast目录改名为itheima；如果itheima目录存在，将itcast目录移动到itheima目录中
```

• 打包压缩命令 tar

作用：对文件进行打包、解包、压缩、解压

语法：

```
tar [-z/-c/-x/-v/-f] fileName [files]
```

> “
>
> i
>
> 包文件后缀为 `.tar` 表示只是完成了打包，并没有压缩
>
> 包文件后缀为 `.tar.gz` 表示打包的同时还进行了压缩

说明：

- `-z` ：z代表的是gzip，通过gzip命令处理文件，gzip可以对文件压缩或者解压
- `-c` ：c代表的是create，即创建新的包文件
- `-x` ：x代表的是extract，实现从包文件中还原文件
- `-v` ：v代表的是verbose，显示命令的执行过程
- `-f` ：f代表的是file，用于指定包文件的名称

举例：

```
#打包
tar -cvf hello.tar ./*      #将当前目录下的所有文件打包，打包后的文件名为hello.tar
tar -zcvf hello.tar.gz ./* #将当前目录下的所有文件打包并压缩，打包后的文件名为
hello.tar.gz

#解包
tar -xvf hello.tar                    #将hello.tar文件进行解包，并将解包后的文件放在
当前目录
tar -zxvf hello.tar.gz                #将hello.tar.gz文件进行解包，并将解包后的文件放
在当前目录
tar -zxvf hello.tar.gz -C /usr/local #将hello.tar.gz文件进行解包，并将解包后的文件放
在/usr/local目录
```

• 文本编辑命令 vi/vim

作用： `vi` 命令是Linux系统提供的一个文本编辑工具，可以对文件内容进行编辑，类似于Windows中的记事本

语法：

```
vi fileName
```

说明：

1. `vim` 是从 `vi` 发展来的一个功能更加强大的文本编辑工具，在编辑文件时可以对文本内容进行着色，方便我们对文件进行编辑处理，所以实际工作中 `vim` 更加常用。
2. 要使用 `vim` 命令，需要我们自己完成安装。可以使用下面的命令来完成安装：

```
yum install vim
```

针对vim中的三种模式说明如下：

1. 命令模式
   - 命令模式下可以查看文件内容、移动光标（上下左右箭头、gg、G）
   - 通过 `vim` 命令打开文件后，默认进入命令模式
   - 另外两种模式需要首先进入命令模式，才能进入彼此
2. 插入模式
   - 插入模式下可以对文件内容进行编辑
   - 在命令模式下按下 `i, a, o` 任意一个，即可进入插入模式。进入插入模式后，下方会出现【INSERT】字样
   - 在插入模式下按下 `ESC` 键，回到命令模式
3. 底行模式
   - 底行模式下可以通过命令对文件内容进行查找、显示行号、退出等操作
   - 在命令模式下按下 `:, /` 任意一个，可以进入底行模式
   - 通过 `/` 方式进入底行模式后，可以对文件内容进行查找
   - 通过 `:` 方式进入底行模式后，可以输入 `wq`（保存并退出）、`q!`（不保存退出）、`set nu`（显示行号）

• 查找命令 find

作用：在指定目录下查找文件

语法：

```
find dirName -option fileName
```

举例：

```
find . -name "*.java"       #在当前目录及其子目录下查找.java结尾文件
find /itcast -name "*.java" #在/itcast目录及其子目录下查找.java结尾的文件
```

• 查找命令 grep

作用：从指定文件中查找指定的文本内容

语法：

```
grep word fileName
```

举例：

```
grep Hello HelloWorld.java #查找HelloWorld.java文件中出现Hello字符串的位置
grep hello *.java          #查找当前目录中所有.java结尾的文件中包含hello字符串的位置
```

• 查看进程 ps

- `ps` 命令是Linux下非常强大的进程查看命令，通过 `ps -ef` 可以查看当前运行的所有进程的详细信息
- “|”在Linux中成为管道符，可以及那个前一个命令的结果输出给后一个命令作为输入
- 使用 `ps` 命令查看进程时，经常配合管道符和查找命令 `grep` 一起使用，来查看特定进程。如查看Tomcat的进程 `ps -ef | grep tomcat`

• 结束进程 kill

语法：

```
kill [-9] 进程id
```

⚠注意：

- `kill` 命令是Linux提供的用于结束进程的命令， `-9` 表示强制结束。例如： `kill -9 7742`

• 防火墙操作

- 查看防火墙状态（ `systemctl status firewalld` 、 `firewall-cmd --state` ）
- 暂时关闭防火墙（ `systemctl stop firewalld` ）
- 永久关闭防火墙（ `systemctl disable firewalld` ）
- 开启防火墙（ `systemctl start firewalld` ）
- 开放指定端口（ `firewall-cmd --zone=public --add-port=8080/tcp --permanent` ）
- 关闭指定端口（ `firewall-cmd --zone=public --remove-port=8080/tcp --permanent` ）
- 立即生效（ `firewall-cmd --reload` ）

- 查看开放的端口（`firewall-cmd --zone=public --list-ports`）

⚠注意：

1. `systemctl` 是管理Linux中服务的命令，可以对服务进行启动、停止、重启、查看状态等操作
2. `firewall-cmd` 是Linux中专门用于控制防火墙的命令
3. 为了保证系统安全，服务器的防火墙不建议关闭

开机启动服务

```
systemctl enable mysqld
```

查看已经启动的服务(net-tools提供的命令)

```
netstat -tunlp
netstat -tunlp | grep mysql
```

rpm 命令

查询当前系统中安装的所有软件

```
rpm -qa
```

举例：

```
rpm -qa | grep mysql    #查询当前系统中安装的名称带mysql的软件
rpm -qa | grep mariadb #查询当前系统中安装的名称带mariadb的软件
```

卸载已安装的软件

```
rpm -e --nodeps 软件名称
```

举例：

```
rpm -e --nodeps mariadb-libs-5.5.60-1.el7_5.x86_64
```

安装rpm软件包

```
rpm -ivh mysql-community-common-5.7.25-1.el7.x86_64.rpm
```

说明：

1. 安装过程中提示缺少net-tools依赖，使用yum安装：`yum install net-tools`
2. 可以通过指令升级现有软件及系统内核：`yum update`

软件安装

软件安装方式

- 二进制发布包安装
  - 软件已经针对具体平台编译打包发布，只要解压，修改配置即可
- rpm安装
  - 软件已经按照redhat的包管理规范进行打包，使用rpm命令进行安装，不能自行解决库依赖问题
- yum安装
  - 一种在线软件安装方式，本质上还是rpm安装，自动下载安装包并安装，安装过程中自动解决库依赖问题
- 源码编译安装
  - 软件以源码工程的形式发布，需要自己编译打包

项目部署

手动部署项目

1. 在IDEA中开发SpringBoot项目并打成jar包
2. 将jar包上传到Linux服务器
3. 启动SpringBoot程序

```
java -jar xxx.jar
```

4. 检查防火墙，确保8080端口对外开放，访问SpringBoot项目
5. 改为后台运行SpringBoot项目，并将日志输出到日志文件

> “
>
> i
>
> `nohup` 命令：英文全称 no hang up (不挂起)，用于不挂断地运行指定命令，退出终端不会影响程序的运行
>
> 语法格式：

```
nohup Command [Arg … ] [&]
```

> 参数说明：
>
> - `Command` ：要执行的命令
> - `Arg` ：一些参数，可以指定输出文件
> - `&` ：让命令在后台运行
>
> 举例：

```
nohup java -jar boot工程.jar &> hello.log & #后台运行java -jar命令，并将日志输出到hello.jar文件
```

6. 停止SpringBoot程序

```
ps -ef | grep 'java -jar'
kill -9 35685
```

• 通过shell脚本自动部署项目

操作步骤：

1. 在Linux中安装Git

```
yum list git #列出git安装包
yum install git #在线安装git
```

2. 在Linux中安装maven

```
tar -zxcf apache-maven-3.5.4-bin.tar.gz -C /usr/local
vim /etc/profile #修改配置文件，加入如下内容
#------------------------------------------------
export MAVEN_HOME=/usr/local/apache-maven-3.5.4
export PATH=$JAVA_HOME/bin:$MAVEN_HOME/bin:$PATH
#------------------------------------------------

source /etc/profile
mvn -version
vim /usr/local/apache-maven-3.5.4/conf/settings.xml #修改配置文件，内容如下
#------------------------------------------------
<localRepository>/usr/local/repo</localRepository>
```

3. 编写Shell脚本（拉取代码、编译、打包、启动）

```
#!/bin/sh
echo ================================
echo  自动化部署脚本启动
echo ================================

echo 停止原来运行中的工程
APP_NAME=helloworld

tpid=`ps -ef|grep $APP_NAME|grep -v grep|grep -v kill|awk '{print $2}'`
if [ ${tpid} ]; then
    echo 'Stop Process...'
    kill -15 $tpid
fi
sleep 2
tpid=`ps -ef|grep $APP_NAME|grep -v grep|grep -v kill|awk '{print $2}'`
if [ ${tpid} ]; then
    echo 'Kill Process!'
    kill -9 $tpid
else
    echo 'Stop Success!'
fi

echo 准备从Git仓库拉取最新代码
cd /usr/local/helloworld

echo 开始从Git仓库拉取最新代码
git pull
echo 代码拉取完成

echo 开始打包
output=`mvn clean package -Dmaven.test.skip=true`

cd target

```

```
echo 启动项目
nohup java -jar helloworld-1.0-SNAPSHOT.jar &> helloworld.log &
echo 项目启动完成
```

4. 为用户授予执行Shell脚本的权限

> “
>
> i
>
> `chmod`（英文全拼：change mode）命令是控制用户对文件的权限的命令
>
> Linux中的权限分别为：读(r)、写(w)、执行(x)三种权限
>
> Linux的文件调用权限分为三级：文件所有者(Owner)、用户组(Group)、其他用户(Other Users)
>
> 只有文件的所有者和超级用户可以修改文件或目录的权限
>
> 要执行Shell脚本需要有对此脚本的执行权限，如果没有则不能执行
>
> ---
>
> `chmod`可以使用八进制数来指定权限：

八进制数	权限	rwx
7	读+写+执行	rwx
6	读+写	rw-
5	读+执行	r-x
4	只读	r--
3	写+执行	-wx
2	只写	-w-
1	只执行	--x
0	无	---

> 举例：

```
chmod 777 bootStart.sh #为所有用户授予读、写、执行权限
chmod 755 bootStart.sh #为文件拥有者授予读、写、执行权限，同组用户和其他用户授予读、执行权限
chmod 210 bootStart.sh #为文件拥有者授予写权限，同组用户授予执行权限，其他用户没有任何权限
```

> ⚠注意：三位数字分别代表不同用户的权限

> - 第1为表示文件拥有者的权限
> - 第2为表示同组用户的权限
> - 第3为表示其他用户的权限

5. 执行Shell脚本

```
sh bootStart.sh
```