# 1 shell 脚本编程概述

## shell 概述

* + 1. Linux 简介
    2. shell 简介
    3. shell的种类和版本

可以通过查看Linux下的相应文件来得知当前Linux中安装的shell种类

# cat /etc/shells

/bin/sh //sh是执行的快捷方式

/bin/dash

/bin/bash

/bin/rbash

bash -version命令查看当前shell的版本号

## 1.2 shell 脚本编程优势

## 1.3 第一个shelll脚本例子

### 1.3.1 shell 脚本的基本元素

whologged.sh:

#！/bin/bash

data #显示日期

who #显示当前的登录用户

#!指定一个文件类型的特殊标记，它告诉linux系统这个文件的执行需要一个解释器。后面

路径指明了解释器在系统的位置。#后面的是注释。

man ls 获取ls命令的帮助信息

### 1.3.2 执行shell脚本

chmod u+x whologged.sh //赋予执行权限

./whologged.sh

# 2 Linux 文件系统和文本编辑器

## 2.1 用户和用户管理

### 2.1.1 用户管理常用命令

用户账号添加命令-----useradd或adduser

useradd [option] [username]

执行该命令的结果是在/etc/passwd文件中增加一行记录，在home目录下创建新用户的主目录。

sudo useradd wang

修改用户账号---usermod

删除用户账号命令----userdel

sudo userdel -r wang

用户口令管理命令----passwd

passwd [option] [username]

sudo passwd wang

sudo tail -l /etc/shadow //查看是否添加成功

## 2.2 文件和目录操作

### 2.2.1 文件操作常用命令

ls 命令

ls -a 显示所有文件，包括隐藏文件

ls -l 显示文件的详细信息

ls -R 递归显示

cp 命令

-a 通常复制目录，保留链接，文件属性，并递归的复制目录

-d 复制时保留链接

-f 删除已经存在的目标文件而不提示

-r 递归复制

-l 不进行复制操作，只是链接文件

mv 命令

rm 命令

### 2.2.2 目录操作常用命令

mkdir命令

-m 对新建目录设置存取权限

-p 可以是一个路径名称。一次可以建立多个目录

mkdir -m 777 tsk

mkdir -p testdir/test

rmdir -p

递归删除目录，删除是目录必须为空

rm -rf 递归的删除目录及目录下文件

cd ~ 返回登录目录

cd - 返回上次访问的目录

pwd 当前目录

chao@ubuntu:~$ cd rk3308\_linux/buildroot/output/

chao@ubuntu:~/rk3308\_linux/buildroot/output$ cd -

/home/chao

### 2.2.3 文件和目录权限管理

chmod命令用于更改文件或目录的访问权限，可以用字母表示法，也可以用数字表示法。

用户类型 数学符号 文件类型

u （user） + 添加某个权限 r 可读

g （group） -取消某个权限 w 可写

o 表示其他用户 = 赋予给定权限并取消其他所有权限 x可执行

a 表示所有用户

数字表示：0没有权限，1可执行权限，2可写权限，4可读权限

**chmod u+x,g+x testvim**

**chmod 764 testvim**

chown 命令 更改文件拥有者命令

-R 递归变更拥有者

### 2.2.4 查找文件命令 find

find 路径 选项 操作

.表示当前目录， /表示系统根目录。可以指定按照文件属主，更改时间，文件类型等条件来

查找

name 根据文件名查找文件

perm 根据文件权限查找文件

depth 查找文件时，先查找当前目录中的文件，然后再其子目录中查找

type 查找某一类型的文件

b 块设备文件

d 目录

c 字符设备文件

p 管道文件

l 符号链接文件

f 普通文件

find命令的操作用于指定结果的输出方式

print 将匹配的文件输出到标准输出

exec 对匹配的文件执行该参数所给出的shell命令。相应的命令的形式为’command’ {} \;注意{} \;空格

**find . -name ‘t\*’ -perm 744 -print //查找以t开头的，且文件属主具有读写执行权限的文件**

**find /etc -type f -name “rc\*” -exec ls -l {} \;**

## 2.3 文本编辑器

命令模式

vim +n file 编辑file文件并将光标置于第n行

vim + file 编辑file文件，并将光标置于最后一行

保存和退出命令

W 保存文本

Q 退出文本编辑器

q! 退出不保存

Wq 保存并退出

插入模式

在命令模式下按I,o,a等字母都可以进入插入模式。ESC进入命令模式

h 将光标向左移动

j将光标向下移动

k将光标向上移动

l将光标向右移动

} 光标移动到段落结尾

{ 光标移动到段落开头

）光标移动到句子的结尾

（ 光标移动到句子的开头

^ 移动到当前行第一个非空字符

$ 移动到当前行末尾

:n 移动到行n

x 删除光标当前位置字符

dd 删除光标在的整行

d$ 删除当前光标位置到改行结束的所有文件

底行工作模式

/ 进入底行工作模式

输入待搜索字符串后按enter开始搜索

vim 配置

**set showmatch:文件中自动显示匹配的括号**

**set nu：在文件中显示行号**

**set autoindent: 编辑时自动缩进**

**set cindent： 按照C语言自动缩进**

# 3 正则表达式

[**Shell脚本中调用另外一个脚本的方法**](https://www.cnblogs.com/royfans/p/7761573.html)

 在Linux平台上开发，经常会在console(控制台)上执行另外一个脚本文件，经常用的方法有：**./my.sh** 或 **source my.sh** 或 **. my.sh**；这三种方法有什么不同呢？我们先来了解一下在一个shell脚本中如何调用另外一个shell脚本，其方法有 **fork    exec    source。**

1、fork  ( /directory/script.sh) ：

如果shell中包含执行命令，那么子命令并不影响父级的命令，在子命令执行完后再执行父级命令。子级的环境变量不会影响到父级。

fork是最普通的, 就是直接在脚本里面用/directory/script.sh来调用script.sh这个脚本. 运行的时候开一个sub-shell执行调用的脚本，sub-shell执行的时候,parent-shell还在。

sub-shell执行完毕后返回parent-shell. sub-shell从parent-shell继承环境变量.但是sub-shell中的环境变量不会带回parent-shell

2、exec (exec /directory/script.sh)：

执行子级的命令后，不再执行父级命令。

exec与fork不同，不需要新开一个sub-shell来执行被调用的脚本.  被调用的脚本与父脚本在同一个shell内执行。但是使用exec调用一个新脚本以后, 父脚本中exec行之后的内容就不会再执行了。这是exec和source的区别

3、source (source /directory/script.sh)：

执行子级命令后继续执行父级命令，同时子级设置的环境变量会影响到父级的环境变量。

与fork的区别是不新开一个sub-shell来执行被调用的脚本，而是在同一个shell中执行. 所以被调用的脚本中声明的变量和环境变量, 都可以在主脚本中得到和使用.

以上三种就是调用shell脚本的不同方法，./my.sh即是fork的方法，source my.sh和. my.sh（点加空格加脚本文件）既是source的方法。

在linux系统上，搭建嵌入式开发平台，在交叉编译代码之前，都需要执行脚本设置环境变量，切记需要使用sourc 或 点的方式执行shell脚本，原因如上。

**嵌入式linux无法执行shell脚本问题**

首先查看/etc/shells文件看支持什么类型的shell,一般是bash

bash -version查看板子的shell版本

编写shell脚本例子：

#/bin/bash //非常重要，指明了解释器路径，没有此句将不能执行bash脚本

int=1

while (( int <= 5 ))

do

echo "$int"

let "int++"

done