day3

mount:挂载，把一个路径挂载到另外一个路径

mkdir mymnt

mount /home/ /mymnt

tar 解压缩

tar –zxvf提取

tar –zcvf压缩

~ 家目录

Ifconfig配置本机的网络地址

Ifconfig 打印所有网络接口的参数

find 查找文件

find ./ -name xxx 从当前路径查找xxx的文件文件夹

find + 路径 –name +文件名

find + 路径 –name +文件名\*

\*通配符，可以出现在任何的位置，匹配任何的字符

rm \* 删除所有文件，统配所有文件并删除

grep 打开某个文档并查找字符串

grep + 要查找的字符串 + 要查找的文件

grep + 要查找的字符串 + 要查找的文件 + -n 显示行号

a在光标后插入

i在光标位置插入

o在光标下一行插入

A在行末插入

I在本行开始插入

O在光标上一行插入

x删除字符（nx删除光标所在处的n个字符）

w 转到下一个单词的开头

b 转到上一个单词的开头

e 转到下一个单词的结尾

gg 定位到文档的开头

ngg 定位到第n行

G定位到文档的结尾

ctrl + v 块操作

shift + v 行操作

SSH Ubuntu没有服务器 只有客户端

linux没有图形界面，很多配置要靠配置文件来配置

MAC可以用自带的命令行客户端登录

SECURE CRT CHARACTER ENCODING 字符编码 改成UTF-8

LOG FILE选中一个文本文档记录

FILE 里面勾选log session日志记录，便于后续的分析

SANBA 以挂载一个网络盘的形式共享文件，在LINUX和WINDOWS之间共享文件

NFS比较简单，NFS太啰嗦了

在终端执行ls pwd ps都是linux的应用程序，命令行中存储的程序，在命令行上执行了一个ls程序，虽然没有.exe后缀，但是也是在linux下的一个可执行程序

这些命令是在执行bin下的可执行文件

环境变量是系统里面的一组变量，它等于某些值，系统定好了一些变量，它对系统来说有特殊用处

如果没给路径名，会在path里一个一个找，以：来分隔

echo $PATH查询环境变量的PATH值

sleep 5前台运行

sleep 5 & 后台运行

kill -9是100%会杀死进程的

脚本是通过调用可执行程序来完成的

脚本负责组合一系列的命令，让它从上往下执行，完成一系列自动化操作，在windows里，脚本叫做批处理，

echo 向屏幕上打印一句话

通过脚本解析器来（shell）执行

#开头的除了#！/bin/bash不是注释之外其他都是注释

#是单行注释

一般脚本写到半个屏幕这么长已经算很长了

执行脚本的时候cd命令不起作用

执行脚本的时候会新开一个终端（脚本解释器），用那个脚本解释器来执行，执行完了之后关掉，对当前终端不影响。

脚本支持条件判断，跳转，读入，读出，参数

read num

echo “Your input num is: $num” echo “This is: ${num}th”

echo 打印有空格的句子一定要加“”，￥取变量

shell脚本里的变量都是字符串，没有数值的概念，没有运算的功能

变量赋值的时候不需要$，引用的时候需要$

echo $?取出上一条命令的退出码，可以看这条命令是否执行成功，如果直接在命令行执行，没什么用，如果这个脚本被其他程序所调用，那个程序可以检查这个脚本的退出码来看这个脚本是否执行成功，exit 0之后结束脚本，后面的代码没用

引用的时候需要$符号

unset str 清空变量str

day4

#!/bin/sh shell脚本解释器

在终端输入 交互式

./执行：重新开一个shell脚本解释器，执行内容，内容执行完，所有都作废

. 执行，把shell脚本的环境引入到当前环境

export 操作环境变量，环境变量是一组变量的值，变量名=XXXXX，环境变量是系统预定义的或者用户定义的，跟终端相关，终端

export AAA=123 创建，修改环境变量

unset AAA删除环境变量

printenv

echo $AAA

比如，脚本判断语言

脚本里面导出的环境变量，./运行在脚本外面是没有用的，. 执行是有用的，把脚本的环境变量引用到当前环境。

“$num”如果没有值，是空的字符串

$sum如果没有值，是空

引用字符串，尽量加双引号，否则输入空格会报错

\*通配符，通配当前目录的所有东西

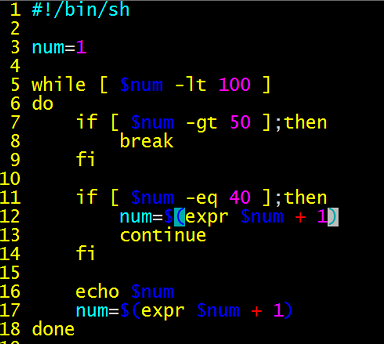
tar -zcvf xxx.tar.gz -C /home指定目录解压

普通文件：可执行文件，文本文件等，不是目录的时候基本上就是普通文件

[ -e /home/if1.sh ] || tar -zcvf shell.tar.gz -C /home

如果[ -e /home/if1.sh ]成立，则执行后面的语句

赋值不需要加$，引用的时候要加$



continue 跳过本轮循环，直接继续进入下一轮循环

break 跳出循环

函数是一个具有特定功能的代码块，每一个模块完成特定功能，最终环境把这些模块组装起来

分摊了任务量，便于调错

隐藏了一部分细节

函数封装一部分代码，函数不会自动执行，函数只能被调用

Shell调用函数时，不需要加括号，参数也不是用括号来传的，有自己的传参方式

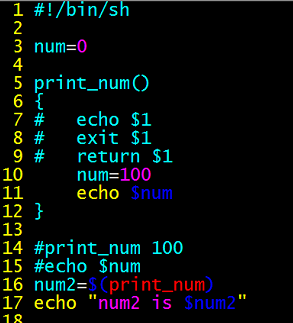
局部变量只能在大括号里面用

& 后台运行

返回值三种方式

return $?

全局变量，函数当中设置一个局部变量，打印出来



直接打印函数，取得函数的时候，函数当中的echo语句没有输出，把值传递给num2，echoecho $(print)

参数扩展，将一串字符串经过替换，扩展，成为另外一串字符串

参数扩展不会改变变量的本身，重新赋值str=${param:-default}会改变

shift移位，对参数，数组进行移位，

行之有效的学习方法

1. 在学新东西之前，找一本中文书，21天学会C语言，C语言从入门到精通，之后通过练习，不断的反馈，把自己不懂的东西搞懂

定向输入和定向输出是针对所有文件用的，不管是程序还是指令

0标准输入，文件描述符

1标准输出，终端的标准输出

wc 单词个数word count

cat hello.txt | wc管道重定向

cat程序的输出变成wc的输入

ps -e | grep sshd

正则表达式比通配表达式更加强大，

ls只认通配表达式，不认正则表达式

正则表达式作为参数给linux的命令使用

grep可以搜多个文件，可以搜带空格的字符串，要加单引号

ls -l | grep ...x..x..x查找可执行程序

反斜杠\：转义 \\* \. \^

sed分为选项和命令，比如：-e 是选项，-e后面的是选项的命令

命令分为两部分，定位命令和编辑命令

再跟上文件名称

diff 比较两个文件不一样的地方

sed -n '/[a-z]\*\/[a-z]\*\//p' profile

[a-z]\*可以代表空

\/代表/，/需要\转义

sed -n '/^#/d' profile 删除注释

sed -n '/^$/d' profile 删除空行

sed -I ‘/unset/i\hello’ profile 将profile当中unset后插入hello

vim当中的替换 s/aaa/bbb 把aaa替换成bbb

保持缓冲区，模式缓冲区

；分号可以隔开命令，两条命令中间加分号两条都能执行