Linux 命令

CD

LS LL

mkdir

rmdir

cp data1.txt data2.txt

rm

pwd

mv

cat

find /home -name "\*.txt"

chmod [who] [操作符] [mode] 文件名

$chmod o+w 　file5

u 表示该档案的拥有者，g 表示与该档案的拥有者属于同一个群体(group)者，o 表示其他以外的人，a 表示这三者皆是。

+ 表示增加权限、- 表示取消权限、= 表示唯一设定权限。

(2)

语法为：chmod abc file

其中a,b,c各为一个数字，分别表示User、Group、及Other的权限。

r=4，w=2，x=1

若要rwx属性则4+2+1=7；

若要rw-属性则4+2=6；

若要r-x属性则4+1=5。

shell 脚本 http://www.runoob.com/linux/linux-shell-variable.html

(第一步$ chmod +x test.sh 添加execute权限

之后一定要写成 ./test.sh，而不是 test.sh)

#!/bin/bash

echo "Hello World !"

->>>>>>>>>>>>变量<<<<<<<<<<<<<<<

for skill in Ada Coffe Action Java; do

echo "I am good at ${skill}Script"

done

->>>>>>>>>>>>$\* 与 $@ 区别：<<<<<<<<<<<<<<<

#!/bin/bash

echo "-- \$\* 演示 ---"

for i in "$\*"; do全部

echo $i

done

echo "-- \$@ 演示 ---"

for i in "$@"; do各个

echo $i

done

->>>>>>>>>数组<<<<<<<<<<<<<<<<<<<<

my\_array=(A B "C" D)

echo "第一个元素为: ${my\_array[0]}"

echo "数组的元素为: ${my\_array[\*]}"

->>>>>>>>>比较+IF+or<<<<<<<<<<<<<<<

if [ $a -lt 100 -o $b -gt 100 ]

then

echo "$a -lt 100 -o $b -gt 100 : 返回 true"

elif condition2

then

command2

else

echo "$a -lt 100 -o $b -gt 100 : 返回 false"

fi

->>>>>>>>>输出 读取+转义+换行+重定向<<<<<<<<<<<<<<<

#!/bin/sh

read name

echo "\"$name It is a test\" \n" > myfile

echo `date` //显示命令执行结果（反引号）

->>>>>>>>>循环<<<<<<<<<<

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字:"

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"

break

;;

esac

done

->>>>>>>>>函数<<<<<<<<<<

funWithReturn(){

echo "这个函数会对输入的两个数字进行相加运算..."

echo "输入第一个数字: "

read aNum

echo "输入第二个数字: "

read anotherNum

echo "两个数字分别为 $aNum 和 $anotherNum !"

return $(($aNum+$anotherNum))

}

funWithReturn

echo "输入的两个数字之和为 $? !"

函数返回值在调用该函数后通过 $? 来获得。

注意：所有函数在使用前必须定义。这意味着必须将函数放在脚本开始部分，直至shell解释器首次发现它时，才可以使用。调用函数仅使用其函数名即可。

funWithParam(){

echo "第一个参数为 $1 !"

echo "第二个参数为 $2 !"

->>>>>>>>>Shell 输入/输出重定向<<<<<<<<<<

如果不希望文件内容被覆盖，可以使用 >> 追加到文件末尾

标准输入文件(stdin)：stdin的文件描述符为0，Unix程序默认从stdin读取数据。

标准输出文件(stdout)：stdout 的文件描述符为1，Unix程序默认向stdout输出数据。

标准错误文件(stderr)：stderr的文件描述符为2，Unix程序会向stderr流中写入错误信息。

#!/bin/bash

exec 1>testopt

exec 2>errlog

如果希望将 stdout 和 stderr 合并后重定向到 file，可以这样写：

$ command > file 2>&1

ES6/ES2015核心内容：https://segmentfault.com/a/1190000004365693

Knockout应用开发指南（完整版） 目录索引：http://www.cnblogs.com/TomXu/archive/2011/11/21/2257154.html

require.js的用法：http://www.ruanyifeng.com/blog/2012/11/require\_js.html

qunit：http://www.bkjia.com/webzh/959840.html

http://www.zhangxinxu.com/wordpress/2013/04/qunit-javascript-unit-test-%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95/

jQuery Ajax：http://www.cnblogs.com/jayleke/archive/2012/08/10/2633174.html

OracleJET

布局（Oracle JET Grids） oj-flex -> oj-flex-item -> oj-md-6 （一共12）

display: large() ? 'all' : 'icons' 图标的展示

组件：ojs/ojcomponent:在Require中率先定义，eg："ojComponent: {component: 'ojButtonset

main-release-paths.json：第三方工具

router

配置

router.configure({

'home': { label: 'Home', value: 'homeContent', isDefault: true },

'book': { label: 'Book', value: 'bookContent' },

'tables': { label: 'Tables', value: 'tablesContent' }

});

子路由

router = oj.Router.rootInstance;

childRouter = router.createChildRouter('chapter');

获取

var orderStateId = this.orderRouter.stateId();

走

1、selection: router.stateId（NAV bar）

2、router.go('home'): transition router to state id 'home'

router.go('/book/chapt2'): transition the root instance to state id 'book' and the child router to state id 'chapt2'

router.go('chapt2/edit'): transition router to state id 'chapt2' and child router to state id 'edit'

组件（按钮）示例

<button id= "button"

data-bind="click: buttonClick,

ojComponent: { component: 'ojButton', label: 'A button element' }">

promise

http://wiki.jikexueyuan.com/project/es6/promise.html

Python

->>>>>>>>>数据类型-集合<<<<<<<<<<

List

L = ['Adam', 95.5, 'Lisa', 85, 'Bart', 59]

print L[-1]、L.append('Paul')

del L[-1] equals L.pop()

tuple（中文翻译为“ 元组 ”。tuple 和 list 非常类似，但是，tuple一旦创建完毕，就不能修改了。）

t = (1,)//正是因为用()定义单元素的tuple有歧义，所以 Python 规定，单元素 tuple 要多加一个逗号“,”

>>> print t

t = ('a', 'b', ['A', 'B'])

L = t[2]

L[0] = 'X'

->>>>>>>>>判断&循环<<<<<<<<<<注意缩进

if age >= 18:

print 'adult'

elif age >= 6:

print 'teenager'

else:

print 'baby'

FOR

L = ['Adam', 'Lisa', 'Bart']

for name in L:

print name

WHILE

while x < 100:

sum += x

x += 2

print sum

->>>>>>>>>数据类型-集合 键值对<<<<<<<<<<

dict 篇

花括号 {} 表示这是一个dict，然后按照 key: value, 写出来即可。最后一个 key: value 的逗号可以省略

d = {

'Adam': 95,

'Lisa': 85,

'Bart': 59

}

>>> len(d)

3

if 'Paul' in d:

print d['Paul']/查成绩

d.get(95)/分数

d['Paul'] = 72/赋值

for key in d:/遍历

print key + ':', d[key]

for v in d.values():

print v

>>> for key, value in d.items():

... print key, ':', value

特点 ：

dict的第一个特点是查找速度快，无论dict有10个元素还是10万个元素，查找速度都一样。而list的查找速度随着元素增加而逐渐下降。

不过dict的查找速度快不是没有代价的，dict的缺点是占用内存大，还会浪费很多内容，list正好相反，占用内存小，但是查找速度慢。

dict是按 key 查找的

dict的第二个特点就是存储的key-value序对是没有顺序的！这和list不一样：

dict的第三个特点是作为 key 的元素必须不可变

set 篇

set 持有一系列元素，这一点和 list 很像，但是set的元素没有重复，而且是无序的，这点和 dict 的 key很像。

>>> s = set(['A', 'B', 'C', 'C'])

>>> print s

set(['A', 'C', 'B'])/去重

>>> len(s)

>>> 'Bart' in s/查找

True

>>> for name in s:/遍历

... print name

>>> s.add(4)

>>> s.remove(4)

特点：

set的内部结构和dict很像，唯一区别是不存储value，因此，判断一个元素是否在set中速度很快。

set存储的元素和dict的key类似，必须是不变对象，因此，任何可变对象是不能放入set中的。

最后，set存储的元素也是没有顺序的。

->>>>>>>>>函数篇<<<<<<<<<<

def quadratic\_equation(a, b, c):

d = b\*b - 4\*a\*c

if d > 0:

return (-b+math.sqrt(d))/(2\*a), (-b-math.sqrt(d))/(2\*a)

elif d == 0:

return (-b)/(2\*a)

else:

return '无解'

print quadratic\_equation(2, 3, 0)

def fn(\*args): //可变参数

print args

->>>>>>>>>切片<<<<<<<<<<

>>> L = ['Adam', 'Lisa', 'Bart', 'Paul']

>>> L[1:3]

['Lisa', 'Bart']

>>> L[::2]

['Adam', 'Bart']

第三个参数表示每N个取一个，上面的 L[::2] 会每两个元素取出一个来，也就是隔一个取一个。

>>> L[-4:-1:2]

['Adam', 'Bart']

记住倒数第一个元素的索引是-1。倒序切片包含起始索引，不包含结束索引。

>>> 'ABCDEFG'[::2]

'ACEG'

->>>>>>>>>列表生成式<<<<<<<<<<

>>> [x \* x for x in range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

->>>>>>>>>python中高级函数<<<<<<<<<<

map()函数

map()是 Python 内置的高阶函数，它接收一个函数 f 和一个 list，并通过把函数 f 依次作用在 list 的每个元素上，得到一个新的 list 并返回。

def f(x):

return x\*x

print map(f, [1, 2, 3, 4, 5, 6, 7, 8, 9])

[1, 4, 9, 10, 25, 36, 49, 64, 81]

reduce()函数

reduce()函数也是Python内置的一个高阶函数。reduce()函数接收的参数和 map()类似，一个函数 f，一个list，但行为和 map()不同，reduce()传入的函数 f 必须接收两个参数，reduce()对list的每个元素反复调用函数f，并返回最终结果值。

filter()函数

filter()函数是 Python 内置的另一个有用的高阶函数，filter()函数接收一个函数 f 和一个list，这个函数 f 的作用是对每个元素进行判断，返回 True或 False，filter()根据判断结果自动过滤掉不符合条件的元素，返回由符合条件元素组成的新list。

sorted()函数

sorted()也是一个高阶函数，它可以接收一个比较函数来实现自定义排序，比较函数的定义是，传入两个待比较的元素 x, y，如果 x 应该排在 y 的前面，返回 -1，如果 x 应该排在 y 的后面，返回 1。如果 x 和 y 相等，返回 0。

->>>>>>>>>python面向对象<<<<<<<<<<

class Person(object):

def \_\_init\_\_(self, name, gender, birth):

self.name = name

self.gender = gender

self.\_\_birth = birth //如果一个属性由双下划线开头(\_\_)，该属性就无法被外部访问

继承

class Student(Person):

def \_\_init\_\_(self, name, gender, score):

super(Student, self).\_\_init\_\_(name, gender)

self.score = score

一定要用 super(Student, self).\_\_init\_\_(name, gender) 去初始化父类，否则，继承自 Person 的 Student 将没有 name 和 gender。

函数super(Student, self)将返回当前类继承的父类，即 Person ，然后调用\_\_init\_\_()方法，注意self参数已在super()中传入，在\_\_init\_\_()中将隐式传递，不需要写出（也不能写）。

RESTful Web 服务应该定义以下方面：

Web 服务的基/根 URI，比如 http://host/<appcontext>/resources。

支持 MIME 类型的响应数据，包括 JSON/XML/ATOM 等等。

服务支持的操作集合（例如 POST、GET、PUT 或 DELETE）。

表 1. RESTful Web 服务示例

方法/资源 资源集合， URI 如：

http://host/<appctx>/resources

GET 列出资源集合的所有成员。

PUT 使用一个集合更新（替换）另一个集合。

POST 在集合中创建数字资源，其 ID 是自动分配的。

DELETE 删除整个资源集合。

开发 REST 服务（https://www.ibm.com/developerworks/cn/web/wa-aj-tomcat/#artdownload）

现在，您已经设置好了开发第一个 REST 服务的环境，该服务对客户端发出 “Hello”。

要做到这一点，您需要将所有的 REST 请求发送到 Jersey 容器 —— 在应用程序的 web.xml 文件中定义 servlet 调度程序（参见清单 1）。除了声明 Jersey servlet 外，它还定义一个初始化参数，指示包含资源的 Java 包。

TestNG与Junit之间的区别（http://blog.csdn.net/luozhuwang/article/details/14646839）

1.对于注解类的支持

2.定义测试任务

3.测试参数化

4.测试依赖性

Solr

Solr\_Home -> web.xml

搜索“天下”

http://127.0.0.1:8080/solr/core1/select?q=%E5%A4%A9%E4%B8%8B&wt=json&indent=true

索引文件：

D:\solr\_home\solr\core1\conf\schema.xml

数据库

D:\solr\_home\solr\core1\conf\data-config.xml

Field

在Solr中，字段(Field)是构成Document的基本单元。对应于数据库表中的某一列。字段是包括了名称，类型以及对字段对应的值如何处理的一种元数据。比如：

<field name="name" type="text\_general" indexed="true" stored="true"/>

Indexed：Indexed=true时，表示字段会加被Sorl处理加入到索引中，只有被索引的字段才能被搜索到。

Stored：Stored=true，字段值会以保存一份原始内容在在索引中，可以被搜索组件组件返回，考虑到性能问题，对于长文本就不适合存储在索引中。

Field Type

Solr中每个字段都有一个对应的字段类型，比如：float、long、double、date、text，Solr提供了丰富字段类型，同时，我们还可以自定义适合自己的数据类型

索引(http://www.importnew.com/12707.html)

Solr/Lucene采用的是一种反向索引，所谓反向索引：就是从关键字到文档的映射过程，保存这种映射这种信息的索引称为反向索引

inverted\_index\_thumb.jpg

左边保存的是字符串序列

右边是字符串的文档（Document）编号链表，称为倒排表（Posting List）

字段串列表和文档编号链表两者构成了一个字典。现在想搜索”lucene”，那么索引直接告诉我们，包含有”lucene”的文档有：2，3，10，35，92，而无需在整个文档库中逐个查找。如果是想搜既包含”lucene”又包含”solr”的文档，那么与之对应的两个倒排表去交集即可获得：3、10、35、92。

SolrJ(http://blog.csdn.net/u012385190/article/details/53115546)