目录

一、 阅读说明

*绿色斜体*代表个人的思考理解，*黄色斜体*代表阅读理解过程中的疑问，红色正体代表关键重要信息，下划线代表次关键重要信息

二、 Linux 帮助命令

1. 帮助文档基本说明

- [] 可选内容
- <> 必选内容
- a | b 二选一
- { } 分组
- ... 同一内容可出现多次

2. whatis

输入“whatis echo”，会显示如下信息。每行包含三部分，第一部分是命令名称；第二部分是命令在 man 手册出现的位置，第三部分是简述命令或函数的作用。

```
1. >>>whatis echo
2. >>>echo (1) - display a line of text
```

如果想详细了解命令信息，可以输入如下命令：

```
1. >>>man 1 echo 或者 man echo(仅当数字为 1 时候可省略，其他数字不可省略)
```

3. 内部命令（builtin）与外部命令

- man bash： NAME 字段后面的命令都是内部命令

4. 查看内部命令使用方法

- help COMMAND 显示 COMMAND 这个命令的用法
- man help 显示所有内部命令列表及使用方法

5. 查看外部命令使用方法

- COMMAND --help

```
1. >>>bash --help //部分命令，也可以简写为 COMMAND -h
```

- man COMMAND　原始格式为：man [章节] COMMAND，默认第 1 章省略

```
1. >>>man bash
```

- info COMMAND

```
1. >>>info bash
```

6. man 手册中字段说明

```
1. NAME 名称及简要说明
2. SYNOPSIS 用法格式说明
```

- [] 可选内容
- <> 必选内容
- a|b 二选一
- { } 分组
- ... 同一内容可出现多次

```
3. DESCRIPTION 详细说明
4. OPTIONS 选项说明
5. EXAMPLES 示例
6. FILES 相关文件
7. AUTHOR 作者
8. COPYRIGHT 版本信息
9. REPORTING BUGS bug 信息
10.   SEE ALSO 其它帮助参考
```

7. man 手册查看方式

- q Q ZZ 退出
- g　1g　　光标跳至文档首部
- G　　光标跳至文档尾部
- e j　　文档前进 N 行
- y k　　文档后退 N 行
- f space 文档前进 N 页
- b ^B　　文档后退 N 页
- /pattern n/N 向后查询、

- ?pattern　　向前查询
- &pattern　　只显示匹配到的行

8. man 手册章节

```
1.  User Commands // 用户命令
2.  System Calls // 系统调用
3.  C Library Functions // C 函数库调用
4.  Devices and Special Files // 设备文件和特殊文件
5.  File Formats and Conventions // 配置文件及格式
6.  Games et. Al. // 游戏
7.  Miscellanea // 杂项
8.  System Administration tools and Deamons // 管理类命令
```

三、 查找命令

1. find

find <指定目录> <指定条件> <指定动作>

- <指定目录>： 所要搜索的目录及其所有子目录。默认为当前目录。

- <指定条件>： 所要搜索的文件的特征。

- <指定动作>： 对搜索结果进行特定的处理。

```
1. find . -name 'my*' -ls
```

2. locate

locate 命令其实是“find -name”的另一种写法，它不搜索具体目录，而是搜索一个数据库（/var/lib/locatedb），这个数据库中含有本地所有文件信息。Linux 系统自动创建这个数据库，每天自动更新一次，所以使用 locate 命令查不到最新变动过的文件。为了避免这种情况，可以先使用 updatedb 命令手动更新数据库再使用 locate。

3. which

which 命令的作用是，在 PATH 变量指定的路径中，搜索某个系统命令的位置，并且返回第一个搜索结果.也就是说，使用 which 命令，就可以看到某个系统命令是否存在，以及执行的到底是哪一个位置的命令.

4. whereis

whereis 命令只能用于程序名的搜索，而且只搜索二进制文件（参数-b）、man 说明文件（参数-m）和源代码文件（参数-s）.如果省略参数，则返回所有信息.

5. type

type 命令其实不能算查找命令，它是用来区分某个命令到底是由 shell 自带的，还是由 shell 外部的独立二进制文件提供的。如果一个命令是外部命令，那么使用-p 参数，会显示该命令的路径，相当于 which 命令.

6. 查看环境变量

env 命令

四、 Linux 根目录结构

bin	存放普通用户可执行的指令	即使在单用户模式下也能够执行处理
boot	开机引导目录	包括 Linux 内核文件与开机所需要的文件
dev	设备目录	所有的硬件设备及周边均放置在这个设备目录中
etc	各种配置文件目录	大部分配置属性均存放在这里
lib/lib64	开机时常用的动态链接库	bin 及 sbin 指令也会调用对应的 lib 库
media	可移除设备挂载目录	类似软盘 U 盘 光盘等临时挂放目录
mnt	用户临时挂载其他的文件系统	额外的设备可挂载在这里,相对临时而言
opt	第三方软件安装目录	现在习惯性的放置在/usr/local 中
proc	虚拟文件系统	通常是内存中的映射,特别注意在误删除数据文件后，比如 DB，只要系统不重启,还是有很大几率能将数据找回来
root	系统管理员主目录	除 root 之外,其他用户均放置在/home 目录下
run	系统运行是所需文件	以前防止在/var/run 中,后来拆分成独立的/run 目录。重启后重新生成对应的目录数据
sbin	只有 root 才能运行的管理指令	跟 bin 类似,但只属于 root 管理员
snap	ubunut 全新软件包管理方式	snap 软件包一般在/snap 这个目录下
srv	服务启动后需要访问的数据目录	
sys	跟 proc 一样虚拟文件系统	记录核心系统硬件信息
tmp	存放临时文件目录	所有用户对该目录均可读写
usr	应用程序放置目录	

- 方法一：DESCRIBE 命令

```
1. DESCRIBE table_name;
```

```
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| bird_id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| scientific_name | varchar(255)| YES  | UNI | NULL    |                |
| common_name     | varchar(50) | YES  |     | NULL    |                |
| family_id       | int(11)     | YES  |     | NULL    |                |
| description     | text        | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

第一列为列名。第二列为类型，第三列用于说明各列能否含有 NULL 值，第四列说明该列是否为键(key)即索引列，第五列用于说明各列默认值，最后一列为备注，额外信息。

- 方法二：SHOW CREATE TABLE 命令

```
1. SHOW CREATE TABLE table_name;
2. SHOW CREATE TABLE table_name \G   //用\G 代替分号，结果将不以表格显示
```

```
*************************** 1. row ***************************
       Table: birds
Create Table: CREATE TABLE `birds` (
  `bird_id` int(11) NOT NULL AUTO_INCREMENT,
  `scientific_name` varchar(255) COLLATE latin1_bin DEFAULT NULL,
  `common_name` varchar(50) COLLATE latin1_bin DEFAULT NULL,
  `family_id` int(11) DEFAULT NULL,
  `description` text COLLATE latin1_bin,
  PRIMARY KEY (`bird_id`),
  UNIQUE KEY `scientific_name` (`scientific_name`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_bin
```

1. 向现有表插入数据

```
1. INSERT INTO birds (scientific_name, common_name)
1. VALUES ('Charadrius vociferus', 'Killdeer'),
2. ('Gavia immer', 'Great Northern Loon'),
3. ('Aix sponsa', 'Wood Duck'),
4. ('Chordeiles minor', 'Common Nighthawk'),
5. ('Sitta carolinensis', ' White-breasted Nuthatch'),
6. ('Apteryx mantelli', 'North Island Brown Kiwi');
```

命令基本格式：**INSERT INTO** table_name (list1, list2) **VALUES** ('Charadrius vociferus', 'Killdeer'),('Gavia immer', 'Great Northern Loon');

2. 查看某一列

```
1. SHOW COLUMNS FROM table_name LIKE 'list_name' \G
```

3. 复制数据表

- 仅复制表结构

```
1. CREATE TABLE table_name_new LIKE table_name
2. INSERT INTO table_name_new SELECT * FROM table_name //从原表复制数据插入
```

- 通过建表实现复制表结构及数据(该做法不会将设定(PRIMARY KEY 等)复制过来)

```
1. CREATE TABLE table_name_new SELECT * FROM table_name //复制全部数据
2. CREATE TABLE table_name_new SELECT list_name1, list_name2 FROM
   table_name //复制其中两列
```

4. 删除数据表

```
1. DROP TABLE table_name
```

5. 更改数据表

更改表的基本语法: ALTER TABLE table_name changes, table_name 为表名，changes 为具体更改命令/更改子句，常见的子句包含：CHANGE/ADD/ALTER/MODIFY/DROP/ORDER BY

- 增加列(在 list_name2 列后增加)

```
1. ALTER TABLE table_name ADD COLUMN list_new INT AFTER list_name2;
2. ALTER TABLE table_name ADD COLUMN list_new INT FIRST  //插入为第一列
```

- 增加多列

```
1. ALTER TABLE table_name
2. ADD COLUMN body_id CHAR(2) AFTER wing_id,
3. ADD COLUMN bill_id CHAR(2) AFTER body_id,
4. ADD COLUMN endangered BIT DEFAULT b'1' AFTER bill_id;
```

- 删除列

```
1. ALTER TABLE table_name DROP COLUMN list_name;
```

- 修改列(非修改表中数据的内容)

```
ALTER TABLE table_name CHANGE COLUMN list_name1 list_name2
VARCHAR(255);
```

如果修改列名，则填入新列名 list_name2，否则仅填入原列名 list_name1 即可，修改类型同理

- 修改列的类型(非前述方法，此方法不可修改列名)

```
ALTER TABLE table_name
MODIFY COLUMN list_name
ENUM('Extinct',
'Extinct in Wild',
'Threatened - Critically Endangered',
'Threatened - Endangered',
'Threatened - Vulnerable',
'Lower Risk - Conservation Dependent',
'Lower Risk - Near Threatened',
  'Lower Risk - Least Concern')
  AFTER family_id;
```

修改列的类型为枚举类型(ENUM)

- 修改表中数据内容

```
UPDATE table_name SET endangered=0 WHERE bird_id IN(1,2,4,5)
```

- 修改默认值

方法 1——采用 CHANGE 子句

```
ALTER TABLE table_name CHANGE COLUMN list_name INT DEFAULT 8;
```

方法二——采用 ALTER 子句

```
ALTER TABLE table_name ALTER list_name SET DEFAULT 7;
```

- 设置 AUTO_INCREMENT 初始值

```
ALTER TABLE table_name AUTO_INCREMENT = 10;
```

Auto_increment 的值来自于 information_schema 数据库中 tables 数据表，tables 表中就有一列叫 auto_increment，新建一个数据表则 tables 表对应增加一行

6. 重命名数据表

```
RENAME TABLE table_name_old TO table_name_new;
```

7. 移动数据表至另一数据库

```
1. RENAME TABLE databse1.table1 TO database2.table2;
```

RENAME 的用法与 linux 中 mv 用法类似，兼具重命名和移动数据表的功能。

8. 重排序数据表

```
1. ALTER TABLE table_name ORDER BY list_name;
```

super() 函数是用于调用父类(超类)的一个方法。单继承：super 与直接用类名调用父类方法无异；多继承，会涉及到查找顺序（MRO）、重复调用（钻石继承）等问题，此时应区分 super 与直接类名调用父类。

MRO 就是类的方法解析顺序表，其实也就是继承父类方法时的顺序表(见附录 1)

9. __str__与__repr__方法

__repr__：repr() 函数将对象转化为供解释器读取的形式，返回一个对象的 string 格式

__str__：str() 函数将对象转化为适于人阅读的形式，返回一个对象的 string 格式，

两个方法均用于返回对象供人阅读，__str__()用于显示给用户，而__repr__()用于显示给开发人员。print 调用的是__str__方法，直接输出实例调用的是__repr__方法

```
2. >>> class Student(object):
3. ... def __init__(self, name):
4. ... self.name = name
5. ...
6. >>> print(Student('Michael'))
7. <__main__.Student object at 0x109afb190>
8. >>> s = Student('Michael')
9. >>> s
10.   <__main__.Student object at 0x109afb310>
```

10.查找模块下的方法(函数)、属性

均在 python 解释器下查询，详见附录 2.查询模块的帮助文档

11.Import 与 from…import…的区别

12.Tuple 与()的使用区别

tuple(seq)，seq -- 要转换为元组的序列。用法如下：

```
1. aList = [123, 'xyz', 'zara', 'abc'];
2. aTuple = tuple(aList)
3. print "Tuple elements : ", aTuple
4.
5. >>>Tuple elements :  (123, 'xyz', 'zara', 'abc')
```

()则直接使用：

```
1. aTuple = (123, 'xyz', 'zara', 'abc')
```

13.Python 时间格式化输出

strftime()函数接收以时间元组(struct_time 对象)，并返回以可读字符串表示的当地时间，格式由参数 format 决定

```
1. t = time.time()//获得以秒为单位的时间
2. print(time.strftime("%b %d %Y %H:%M:%S", time.gmtime(t)))//gmtime 获得 struct_time 对象
```

格式参数 format，详见附录 3.python 中时间日期格式化符号

14.格式化输出

格式输出详见参考文档 《python 的格式化输出》

15.Windows 环境下文件路径表示

Python 代码里面，反斜杠“\”是转义符，例如“\n”表示回车，采用以下三种方式表示路径：

- 斜杠 “/”，如“c:/test.txt”
- 两个反斜杠 “\\”，如“c:\\test.txt”
- 字符串前面加上字母 r，表示后面是一个原始字符串 raw string，如“r“c:\\test.txt””

16.Python 接收命令行参数

导入 argv，结果即为参数列表

```
1. from sys import argv
2. print(argv)
3. >>>python xx.py xxx
4. >>>['xx.py','xxx']
```

17.'\u'前缀字符串

\u4f60 十六进制代表对应汉字的 utf-16 编码

18.定义 1 个元素的 tuple

定义 1 个元素的 tuple: t = (1,)，加上一个逗号，避免成为数学意义上的括号

19.创建生成器(generator)的两种方式

方式 1: g = (x * x for x in range(10))，列表生成式的[]更改为()

方式 2: 如果一个函数定义中包含 yield 关键字，那么函数就不再是一个普通函数，而是一个 generator，函数是顺序执行，遇到 return 语句或者最后一行函数语句返回。而 generator 在每次调用 next()的时候执行，遇到 yield 语句返回，再次执行时从上次返回的 yield 语句处继续执行

```
1. def fib(max):
2.     n, a, b = 0, 0, 1
3.     while n < max:
4.         yield b
5.         a, b = b, a + b
6.         n = n + 1
7.     return 'done'
```

20.python 代码规范

请参考《python 代码规范》

21.迭代器总结

- 凡是可作用于 for 循环的对象都是 Iterable 类型；
- 凡是可作用于 next()函数的对象都是 Iterator 类型，它们表示一个惰性计算的序列；
- 集合数据类型如 list、dict、str 等是 Iterable 但不是 Iterator，不过可以通过 iter()函数获得一个 Iterator 对象。
- Python 的 for 循环本质上就是通过不断调用 next()函数实现的

22.Map 函数总结

map()函数接收两个参数，一个是函数，一个是 Iterable，map 将传入的函数依次作用到序列的每个元素，返回新的 Iterator。

```
map(function, iterable1,iterable2 ...)
```

例如三个列表相乘：

```
1. list1 = [1,2,3,4,5,6,7,8,9]
```

```
list2 = [1,2,3,4,5,6,7,8,9]
list3 = [9,8,7,6,5,4,3,2,1]
def foo(l1,l2,l3):
    return l1*l2*l3

print(list(map(foo,list1,list2,list3)))
>>>[9,32,63,96,125,144,147,128,81]
```

23.Reduce 函数总结

reduce 把一个函数作用在一个序列[x1, x2, x3, ...]上，这个函数必须接收两个参数，reduce 把结果继续和序列的下一个元素做累积计算。

例如序列求和：

```
>>> from functools import reduce
>>> def add(x, y):
...     return x + y
...
>>> reduce(add, [1, 3, 5, 7, 9])
25
```

24.匿名函数(lambda)总结

lambda parameters: expression

parameters：可选，如果提供，通常是逗号分隔的变量表达式形式，即位置参数。

expression：**不能包含分支或循环（但允许条件表达式）**，也不能包含 return（或 yield）函数。如果为元组，则应用圆括号将其包含起来。调用 lambda 函数，**返回的结果是对表达式计算产生的结果**

```
#根据参数是否为 1 决定 s 为 yes 还是 no
>>> s = lambda x:"yes" if x==1 else "no"
```

25.全局变量的使用

使用到的全局变量只是作为引用，不在函数中修改它的值的话，不需要加 global 关键字

```
a = 1

def func():
```

```
4.     if a == 1:
5.         print("a: %d" %a)
```

使用到的全局变量，需要在函数中修改的话，就涉及到歧义问题，因此，需要修改全局变量 a，可以在"a = 2"之前加入 global a 声明

```
1. a = 1
2.
3. def func():
4.     global a
5.     a = 2
6.     print("in func a:", a)
```

26.python 定义类时，内部方法的互相调用

每次调用内部的方法时，方法前面加 self

```
1. class MyClass:
2.     def __init__(self):
3.         pass
4.     def func1(self):
5.         print('a')
6.         self.common_func()
7.      def func2(self):
8.         self.common_func()
9.
10.     def common_func(self):
11.         pass
```

27.filter 函数总结

filter()也接收一个函数和一个序列。filter()把传入的函数依次作用于每个元素，然后根据返回值是 True 还是 False 决定保留还是丢弃该元素

28.sorted 函数总结

sorted()函数也是一个高阶函数，它还可以接收一个 key 函数来实现自定义的排序，key 指定的函数将作用于 list 的每一个元素上，并根据 key 函数返回的结果进行排序

```
1. >>> sorted([36, 5, -12, 9, -21], key=abs)
2. [5, 9, -12, -21, 36]
```

29.返回函数(闭包)

30.

31.

32.装饰器

这种在代码运行期间动态增加功能的方式，称之为“装饰器”（Decorator）。本质上，decorator 就是一个返回函数的高阶函数。

```
def log(func):
    def wrapper(*args, **kw):
        print('call %s():' % func.__name__)
        return func(*args, **kw)
    return wrapper
```

如果将上述的 log 函数作为装饰器，则在其装饰的函数前添加@log

```
@log
def now():
    print('2015-3-25')
>>> now()
call now():
2015-3-25
```

装饰器原理及引入机制较为复杂，详情可参考《python 装饰器解释》

33.函数参数

- 必选参数(位置参数)：
- 默认参数：如 def power(x, n=2)，x 即为位置参数，n 为默认参数，必选参数在前，默认参数在后
- 可变参数：传入的参数个数是可变的，不必自行将所有参数组装成一个 list 或 tuple，如 def calc(*numbers)。定义可变参数和定义一个 list 或 tuple 参数相比，仅仅在参数前面加了一个*号。在函数内部，参数 numbers 接收到的是一个 tuple

```
def calc(*numbers):
    sum = 0
    for n in numbers:
        sum = sum + n * n
    return sum
```

- 关键字参数:可变参数在函数调用时自动组装为一个 tuple。而关键字参数允许你传入 0 个或任意个含参数名的参数，这些关键字参数在函数内部自动组装为一个 dict:

```
1. def person(name, age, **kw):
2.     print('name:', name, 'age:', age, 'other:', kw)
3.
4. >>> person('Michael', 30)
5. name: Michael age: 30 other: {}
6. >>> person('Bob', 35, city='Beijing')
7. name: Bob age: 35 other: {'city': 'Beijing'}
```

34.字符串里面的引号

单引号''定义字符串的时候，它就会认为你字符串里面的双引号" "是普通字符，从而不需要转义。反之当你用双引号定义字符串的时候，就会认为你字符串里面的单引号是普通字符无需转义

```
1. Str1 = "We all know that 'A' and 'B' are two capital letters."
2. Str2 = 'The teacher said: "Practice makes perfect" is a very famous
   proverb.'
```

35.偏函数

functools.partial 的作用就是，把一个函数的某些参数给固定住（也就是设置默认值），返回一个新的函数，调用这个新函数会更简单

```
1. >>> import functools
2. >>> int2 = functools.partial(int, base=2)
3. >>> int2('1000000')
4. 64
```

36.算法速度的定义

- 大 O：T(N) = O(f(N))，T 增长率小于等于 f
- Ω：T(N) = Ω (f(N))，T 增长率大于 f
- Θ：T(N) = Θ (f(N))，T 增长率等于 f
- 小 o：T(N) =o (f(N))，T 增长率小于 f

37.计算运行时间的一般法则

- 法则 1：for 循环：一个 for 循环的运行时间至多是该循环内语句的运行时间乘以迭代次数

- 法则 2：嵌套 for 循环：嵌套循环内部的一条语句总运行时间为该语句运行时间乘以所有 for 循环的大小，如下程序片段运行时间为 $O(N^2)$

```
for i in range(N):
    for j in range(N):
        j += 1
```

- 法则 3：顺序语句：各个语句运行时间求和
- 法则 4：if/else 语句：运行时间至多是判断时间+S1 或 S2 中运行时间长者

```
if (condition):
    S1
else:
    S2
```

38.导入自定义模块

39.子类继承父类的属性问题

更加细致的继承中的属性和方法问题，请参考《python 类的继承、属性总结和方法总结》

如果子类自己定义了__init__方法，那么父类的属性是不能调用的，如下：

```
class Animal:
    def __init__(self):
        self.a = 'aaa'

class Cat(Animal):
    def __init__(self):
        pass

cat = Cat()
print(cat.a)

>>>AttributeError: 'Cat' object has no attribute 'a'
```

可以在子类的 __init__中调用一下父类的 __init__ 方法,这样就可以调用父类的属性

```
class Animal:
```

```
2.      def __init__(self):
3.          self.a = 'aaa'
4.
5. class Cat(Animal):
6.      def __init__(self):
7.          super().__init__()
8.
9. cat = Cat()
10.    print(cat.a)
11.
12.    >>>aaa
```

40.子类扩展父类属性

41.附录

1. 方法解析顺序（Method Resolution Order, MRO）列表

MRO 列表的顺序遵循以下三条原则：

- 子类永远在父类前面
- 如果有多个父类，会根据它们在列表中的顺序被检查
- 如果对下一个类存在两个合法的选择，选择第一个父类

7. 查询模块的帮助文档

- 先导入模块，再查询普通模块的使用方法：help(module_name)，例：help(math)
- 先导入 sys，再查询系统内置模块的使用方法：sys.bultin_modulenames
- 查看模块下所有函数：dir(module_name)，例：dir(math)
- 查看模块下特定函数：help(module_name.func_name)，例：help(math.sin)
- 查看函数信息的另一种方法：print(func_name.__doc__)，例：print(sin.__doc__)

8. python 中时间日期格式化符号

- %y 两位数的年份表示（00-99）
- %Y 四位数的年份表示（000-9999）
- %m 月份（01-12）
- %d 月内中的一天（0-31）
- %H 24 小时制小时数（0-23）
- %I 12 小时制小时数（01-12）

- %M 分钟数（00=59）
- %S 秒（00-59）
- %a 本地简化星期名称
- %A 本地完整星期名称
- %b 本地简化的月份名称
- %B 本地完整的月份名称
- %c 本地相应的日期表示和时间表示
- %j 年内的一天（001-366）
- %p 本地 A.M.或 P.M.的等价符
- %U 一年中的星期数（00-53）星期天为星期的开始
- %w 星期（0-6），星期天为星期的开始
- %W 一年中的星期数（00-53）星期一为星期的开始
- %x 本地相应的日期表示
- %X 本地相应的时间表示
- %Z 当前时区的名称
- %% %号本身