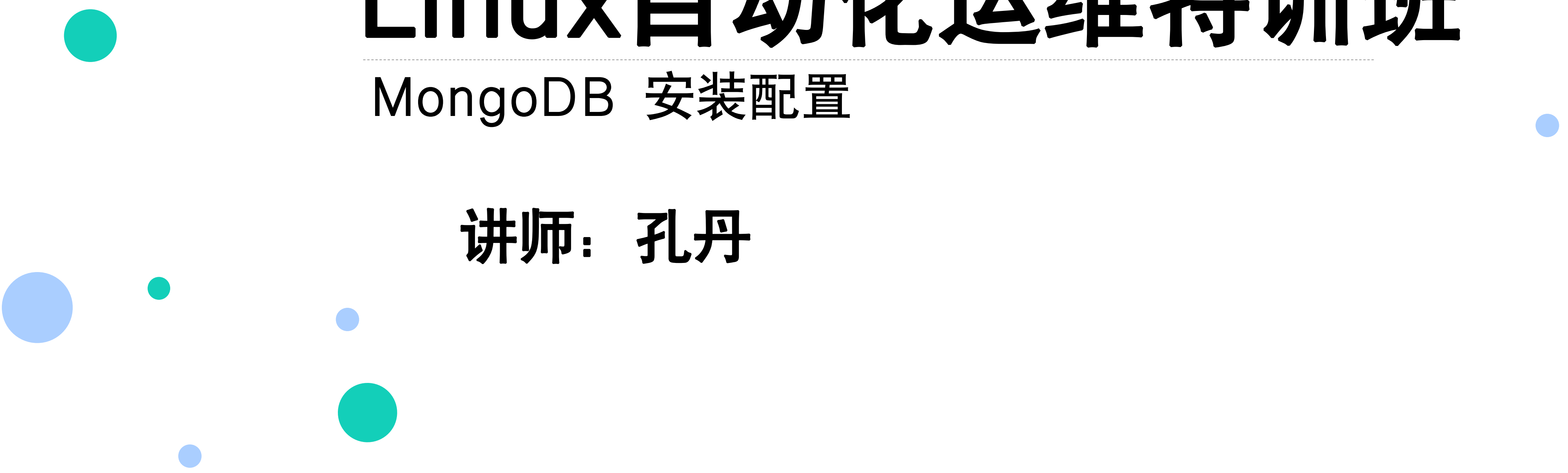




# Linux自动化运维特训班

正则表达式

讲师：孔丹



# 大纲

- 什么是正则表达式：主要介绍什么是正则表达式，为什么要学习正则表达式以及如何学习和实践正则表达式。
- 正则表达式基础：主要介绍正则表达式的元字符、扩展元字符，如何匹配单个字符，如何匹配多个字符，如何匹配字符串的开头或者结尾以及运算符的优先级。
- grep命令：主要介绍grep命令的基本语法，正则表达式在grep命令中的应用以及grep命令族中的其他的命令简介。

# 什么是正则表达式

- 所谓正则表达式，实际上就是用来描述某些字符串匹配规则的工具。由于正则表达式语法简练，功能强大，得到了许多程序设计语言的支持，包括Java、C++、Perl以及Shell等。对于初学者来说，首次接触正则表达式非常难以接受，本节将介绍正则表达式的入门知识，以利于后面几节的学习。

# 为什么使用正则表达式

- 在进行程序设计的过程中，用户会不可避免地遇到处理某些文本的情况。有的时候，用户还需要查找符合某些比较复杂规则的字符串。对于这些情况，如果单纯依靠程序设计语言本身，则往往会使得用户通过复杂的代码来实现。但是，如果使用正则表达式，则会以非常简短的代码来完成。

# 如何学习正则表达式

- 1. 重点在于理解元字符
- 2. 掌握好正则表达式的语法
- 3. 开拓思路，寻找最佳的表达方法

# 如何实践正则表达式

□ 当一个正则表达式完成之后，并能够保证这个表达式一定是准确的，需要不断地测试才可以确定其正确与否。在不同的环境下，用户需要不同的工具来帮助他完成测试的过程。如果是在Shell命令行中，用户可以使用grep命令来测试。

□ grep家族有三大成员分别为：

grep：支持使用基本正则表达式。

egrep：支持使用扩展正则表达式。

fgrep：不支持使用正则表达式。

□ grep命令：

功能：根据用户指定的“pattern（过滤条件）”“对目标文本逐行进行匹配检查；打印出符合条件的行，即文本搜索工具。注：PATTERN即过滤条件指由文本字符及正则表达式元字符所编写的字符串。

# 如何实践正则表达式

□grep命令:

参数

-n :显示行号

-o :只显示匹配的内容

-q :静默模式, 没有任何输出, 得用\$?来判断执行成功没有, 即有  
没有过滤到想要的内容

-l : 如果匹配成功, 则只将文件名打印出来, 失败则不打印, 通常-  
rl一起用, grep -rl 'root' /etc

-A :如果匹配成功, 则将匹配行及其后n行一起打印出来

-B :如果匹配成功, 则将匹配行及其前n行一起打印出来

-C :如果匹配成功, 则将匹配行及其前后n行一起打印出来

--color

-c :如果匹配成功, 则将匹配到的行数打印出来

-E :等于egrep, 扩展

-i : 忽略大小写

-v :取反, 不匹配

-w: 匹配单词



# 基本正则表达式

□ 基本正则表达式（Basic Regular Expression, BRE），又称为标准正则表达式，是最早制订的正则表达式规范，仅支持最基本的元字符集。基本正则表达式是POSIX规范制订的两种正则表达式语法标准之一，另外一种语法标准称为扩展正则表达式，将在随后介绍。

字符	含义
^	在每行的开始进行匹配
\$	在每行的末尾进行匹配
\<	在字的开始进行匹配
\>	在字的末尾进行匹配
.	对任何单个字符进行匹配
[str]	对str中的任何单个字符进行匹配
[^str]	对任何不在str中的单个字符进行匹配
[a-b]	对a到b之间的任何字符进行匹配
\	抑止后面的一个字符的特殊含义
*	对前一项(item)进行0次或多次重复匹配



# 基本正则表达式

## □ 基础正则表达式: BRE (basic regular expression)

- 1) ^word 表示搜索以word开头的内容。
- 2) word\$ 表示搜索以word结尾的内容。
- 3) ^\$ 表示空行, 不是空格。
- 4) . 代表且只能代表一个任意字符。
- 5) \ 转义字符, 让有着特殊身份意义的字符。  
例如: \.只表示小数点, 还原原始的小数点的意义。
- 6) \* 重复0个或多个前面的字符
- 7) .\* 匹配所有的字符。^. \* 任意多个字符开头。
- 8) [] 匹配字符集合内任意一个字符, 如[a-z]
- 9) [^abc] ^在中括号里表示非, 不包含a或b或c
- 10) {n,m} 匹配n到m次, 前一个字符。

{n,} 至少N次, 多了不限。

{n} N次

{,m} 至多m次, 少了不限。

注意: grep要{转义}, \{\}, egrep不需要转义

11) \<或\b: 锚定词首(支持vi和grep), 其后面的任意字符必须作为单词首部出现, 如 \

12) \>或\b: 锚定词尾(支持vi和grep), 其前面的任意字符必须作为单词尾部出现, 如 love\>或love\b

# 扩展正则表达式

- 扩展正则表达式 (Extended Regular Expression, ERE) 支持比基本正则表达式更多的元字符，但是扩展正则表达式对有些基本正则表达式所支持的元字符并不支持。前面介绍的元字符“^”、“\$”、“.”、“\*”、“[]”以及“[^]”这6个元字符在扩展正则表达式都得到了支持，并且其意义和用法都完全相同，不再重复介绍。接下来重点介绍一下在扩展正则表达式中新增加的一些元字符。

字符	含义
+	对前一项进行1次或多次重复匹配
?	对前一项进行0次或1次重复匹配
{j}	对前一项进行j次重复匹配
{j,}	对前一项进行j次或更多次重复匹配
{,k}	对前一项最多进行k次重复匹配
{j,k}	对前一项进行j到k次重复匹配
s t	匹配s项或t项中的一项
(exp)	将exp作为单项处理

# 正则表达式

□ 后项引用或者称作分组引用

\( \)

\(ab\)\*

后项引用

\1: 引用第一个左括号以及与之对应的右括号所包括的所有内容

\2:

\3:

示例：显示/etc/inittab文件中以一个数字开头并以一个与开头数字相同的数字结尾的行；

egrep '^([0-9]).\*\1\$' /etc/inittab

# 正则表达式案例

- 1、显示/etc/passwd文件中以bash结尾的行；
- 2、找出/etc/passwd文件中的三位或四位数；
- 3、找出/etc/grub2.cfg文件中，以至少一个空白字符开头，后面又跟了非空白字符的行
- 4、找出"netstat -tan"命令的结果中，以`LISTEN`后跟0或多个空白字符结尾的行；
- 5、找出"fdisk -l"命令的结果中，包含以/dev/后跟sd或hd及一个字母的行；
- 6、找出"ldd /usr/bin/cat"命令的结果中文件路径；
- 7、找出/proc/meminfo文件中，所有以大写或小写s开头的行；至少用三种方式实现；
- 8、显示当前系统上root、centos或spark用户的相关信息；
- 9、echo输出一个绝对路径，使用egrep取出其基名；
- 10、找出ifconfig命令结果中的1-255之间的整数；
- 11、找出系统中其用户名与shell名相同的用户。

# 总结

- 基本正则表达式

- 扩展正则表达式



# 作业

- ❑ 1、显示/etc/rc.d/rc.sysinit文件中以不区分大小的h开头的行；
- ❑ 2、显示/etc/passwd中以sh结尾的行；
- ❑ 3、显示/etc/fstab中以#开头，且后面跟一个或多个空白字符，而后又跟了任意非空白字符的行；
- ❑ 4、查找/etc/rc.d/rc.local中包含“以to开始并以to结尾”的字串行；
- ❑ 5、查找/etc/inittab中含有“以s开头，并以d结尾的单词”模式的行；
- ❑ 6、查找ifconfig命令结果中的pv4地址；
- ❑ 7、显示/var/log/secure文件中包含“Failed”或“FAILED”的行
- ❑ 8、在/etc/passwd中取出默认shell为bash
- ❑ 9、以长格式列出/etc/目录下以ns开头、.conf结尾的文件信息
- ❑ 10、高亮显示passwd文件中冒号，及其两侧的字符
- ❑ 11、匹配/etc/services中开头结尾字母一样的单词





# 谢谢观看

更多好课，请关注[万门大学APP](#)

