###### 回顾

1. 变量类别：本地、环境、局部、位置、特殊
2. Bash配置文件：

全局：/etc/profile, /etc/profile.d/\*.sh, /etc/bashrc

用户：~/.bash\_profile, ~/.bashrc

1. 计算机理解：二进制代码

程序 = 指令（取决于芯片种类）+ 数据（存储器，IO设备）

1. 编程语言：编译型语言（编译器）VS 解释型语言（解释器）

问题空间🡪解空间

1. Bash编程：
   1. 指令：bash所在OS上的可运行指令
   2. 翻译：在当前OS上查找相应命令并提交给内核的执行的过程
   3. 程序控制语句：顺序、循环、选择

###### Linux文本搜索/查找

1. 文件搜索globbing：p\*d，表示之间可出现任意长度的字符或字符串
2. Grep，egrep，fgrep：
   1. Grep：Global search Regular expression and Print out the line，全面搜索正则表达式并输出
   2. 文本搜索工具，根据用户指定的文本模式（搜索条件）对目标文件进行逐行搜索，显示能匹配到的行
   3. 无法对文件名进行搜索
   4. 语法格式：grep [option]… ‘PATTERN’(模式，搜索条件) FILE…

Grep ‘root’ /etc/passwd

[option]:--color=auto,查找的字符或字符串用不同颜色突出表示

1. 正则表达式：
   1. 一类字符所书写的模式，其中许多字符不表示其字面意义，而是表达控制或通配等功能；

元字符：不表示其字面意义，而用于额外功能性描述

* 1. 正则表达式引擎：理解正则表达式意义，由程序实现
  2. 基本正则表达式：表达功能较弱，grep支持
  3. 扩展正则表达式：表达功能较强，egrep支持（e：extended，相当于grep -E）
  4. Fgrep：fast，不支持使用正则表达式，直接按字符表面意义进行搜索

1. 基本正则表达式的元字符：
   1. 字符匹配：
2. ‘.’——匹配任意单个字符
3. ‘[]’——匹配指定范围内的任意单个字符

[0-9],[[:digit:]]； [a-z],[[:lower:]]； [A-Z],[[:upper:]]；

[[:space:]]（空格）； [[:punct:]]（标点符号）；

[[:alpha:]]（字母）； [[:alnum:]]（字母和字符）；

1. [^]——取反
   1. 次数匹配：
2. 用于实现指定其前面的字符所能够出现的次数
3. ‘\*’——它前面的字符可以出现任意次数：x\*y
4. ’\?’——（\表示转义）它前面的字符是可有可无的：x\?y:ay
5. ‘\+’——至少出现一次
6. ‘\{m\}’——（\表示转义）它前面的字符要出现m次：x\{2\}y:xxxy
7. ‘\{m,n\}’——（\表示转义）它前面的字符出现至少m次，至多n次：x\{2,5\}y:xxy
8. ‘\{m,\}’——（\表示转义）它前面的字符出现至少m次
9. ‘\{0,n\}’——（\表示转义）它前面的字符出现至多n次
10. ‘.\*’——任意长度的任意字符，工作于贪婪模式，及尽可能多地去匹配
11. 位置锚定：
    1. ‘^’——行首锚定，匹配串必须出现在行首，写在模式最左侧
    2. ‘$’—— 行尾锚定，匹配串必须出现在行尾，写在模式最右侧
    3. ‘^$’—— 空白行：grep “^$” /etc/issue | wc -l (统计出现多少行)
    4. ‘\<’——词首锚定，要出现在单词左侧：grep “\<r..t” /etc/passwd
    5. ‘\>’——词尾锚定，要出现在单词右侧：grep “r..t\>” /etc/passwd
    6. ‘\b’——词首词尾都可以锚定，出现在单词左侧或右侧
    7. ‘\<word\>’——特殊单词锚定：grep “\<r..t\>” /etc/passwd
12. 分组
    1. ‘\(\)’:例如：grep “\(ro\)\*” /etc/passwd——出现任意次

grep “\(ro\)\{1,\}” /etc/passwd——出现至少1次

* 1. 分组中的模式匹配到的内容，可由正则表达式引擎记忆在内存中，之后可被多次引用：例如\(ab\(x\)y\).\*\(mn\),括号有编号，自左而右的左括号，以及与其匹配右括号，自内向外匹配，自左向右编号
  2. ‘\#’：引用第n个括号所匹配到的内容，而非模式本身

例如: \(ab\?c\).\*\1:abcmnaaa，abcmnac不匹配,abcmnabc,acmnac匹配

1. 命令选项：
   1. ‘-v’——反向选取：grep –v “^$” /etc/passwd，所有非空白行
   2. ‘-O’——仅显示匹配到的字串本身，而非字串所在行：grep –O “r..t” /etc/passwd
   3. ‘–i’——ignore case，忽略字符大小写：[Cc]at != -i “cat”(at也不区分大小写)
   4. “ –E”——支持使用扩展正则表达式
   5. “ –A #”——显示当行和后面n行
   6. “ –B #”——显示当行和前面n行
   7. “ –C #”——显示当行和上下文，即前后n行

###### 练习题

1. 显示/proc/meminfo文件中以大写或小写s开头的行
2. 显示/etc/passwd文件中以其默认shell为非/sbin/nologin的用户
3. 显示/etc/passwd文件中以其默认shell为/bin/bash的用户，进一步显示上述结果中其ID号最大的用户
4. 找出/etc/passwd/文件中的一位数或两位
5. 显示/boot/grub/grub.conf中至少一个空白字符开头的行
6. 显示/etc/rc.d/rc.sysinit文件中，以#开头，后面跟至少一个空白字符，而后又有至少一个非空白字符的行
7. 找出netstat –tan命令执行结果中以‘LISTEN’结尾的行
8. 添加用户bash，testbash，basher，nologin(SHELL为/sbin/nologin)而找出当前系统上其用户名和默认shell相同的用户
9. 扩展题：新建一个文件夹，假设有如下内容：

He like his lover.

He love his lover.

He like his liker.

He love his liker.

找出其中最后一个单词是由此前某单词加r构成的行。

Grep “\(l..e\).\*\1r” a.txt

###### Answer

1. Grep –i ‘^s’ /proc/meminfo; Grep ‘^[Ss]’ /proc/meminfo；

Grep –E ‘^(S|s)’ /proc/meminfo

1. Grep –v “/sbin/nologin$” /etc/passwd | cut –d: -f1
2. Grep “/bin/bash$” /etc/passwd |sort –t: -k3 –n | tail –l | cut –d: -f1
3. Grep “\<[0-9][0-9]\?\>” /etc/passwd; Grep “\<[0-9]\{1,2\}\>” /etc/passwd
4. Grep “^[[:space:]]\{1,\}” /boot/grub/grub.conf
5. Grep “^#[[:space:]]\{1,\}[^[:space:]]\{1,\}” /etc/rc.d/rc.sysinit
6. Netstat –tan | grep “LISTEN[[:space:]]\*$”
7. Useradd bash

Useradd testbash

Useradd basher

Useradd –s /sbin/nologin nologin

Grep “^\([[:alnum:]]\{1,\}\):.\*\1$” /etc/passwd