Bash

· 数值比较

- test 的数值比较功能
  - * n1 -eq n2 : 相等
  - * n1 -ge n2 : ≥
  - * n1 -gt n2 : >
  - * n1 -le n2 : ≤
  - * n1 -lt n2 : <
  - * n1 -ne n2 : 不≠

⇒ # E.g.:

方括号等于 test <condition>

```
if [$value1 -gt 5]
then
    echo "success"
fi
```

· 字符串比较

- test 的字符串比较测试
  - * str1 = str2
  - * str1 != str2
  - * str1 < str2 （必须转义）
  - * str1 > str2
  - * -n str1 : 检测是否非0
  - * -z str2 : 检测是否为0
- 在比较中，大写字母认为是小于小写字母的。这是因为使用 ASCII 顺序，而 sort 等命令使用本地化设置中定义的顺序排序（英语），默认为大写字母大于小写字母。

· 文件比较

- test 的文件比较功能
  - * -d file : file是否存在并是一个目录
  - * -e file : file是否存在
  - * -f file : file是否存在并是一个文件
  - * -r file : file是否存在并可读
  - * -s file : file是否存在并非空
  - * -w file : file是否存在并可写
  - * -x file : file是否存在并可执行
  - * -O file : file是否存在并属当前用户所有
  - * -G file : file是否存在并且默认组与当前用户相同
  - * file1 -nt file2 : file1是否比file2新
  - * file1 -ot file2 : file1是否比file2旧

· 复合条件测试：

```
[ condition1 ] && [condition2]
[condition1] || [Condition2]
```

· case 命令

```
case variable in
pattern1 | pattern2) commands1 ;;
pattern3) commands2
*) default commands ;;
esac
```

- C语言风格的for命令：

```
for (( i=1; i <= 10; i++ ))
```

或使用多个变量：

```
for (( a=1, b=10; a <= 10; a++, b-- ))
```

- 可以在while和until循环中加入多个test，但仅有最后一个状态码决定是否执行代码
- 3个特殊变量处理命令行参数：
  - $#：命令行输入的参数总数
  - $*：将所有参数保存为一个字符串
  - $@：将所有变量保存为单独的词
- 重定向符号：
  - 2>：STDERR重定向
  - &>：STDERR和STDOUT重定向为一个文件，STDERR有更高的优先级
  - >：STDOUT重定向
  - >&2：将脚本中单独一行重定向到STDERR（echo "error" >&2）
  - 可以用exec重定向为STDOUT或STDERR（exec 2>testerror），当然，也可以重定向STDIN（exec 0< testfile）；exec可以使用除0，1，2以外（3-8）剩下文件描述符，如 exec 3> testout，然后在文本中可定义输出到该文件 echo "hi" >&3
  - 可以手动关闭重定向的文件描述符，即将其重定向为&-（exec 3>&-）。
  - 可以用lsof查看文件描述符（lsof -a -p $$ -d 0,1,2）。
  - 可以将STDERR重定向到/dev/null，也可将其作为输入；该文件不包含任何数据。
- 创建本地临时文件：（此时，文件名为全路径）
  - 可以用 mktemp test.XXXXXX 来创建临时文件，其中 -t 可以在临时目录创建文件，-d 可以用来创建临时目录。
- 可以用tee命令同时把结果输出到STDOUT和指定位置，选项 -a 可以将结果追加到文件中。

· sed:

- "s" 用来替换文本：默认只会替换第一处出现的文本，所以可以用不同的替换标记：

  's/test/trial/flags'

  将 test 替换成 trial，其中 flags 可选：

  * 数字，表明替换第几处出现的地方。
  * g，表明替换所有。
  * p，表明原先的内容要打印出来。
  * w file，将替换后的结果写入文件中。

- "d" 用来删除行

'3d' 文件结尾字符	删除第3行
'3,$d'	删除第3行开始到结尾的所有行
'/number 1/d'	删除含有 "number 1" 的所有行。
'1!,15!d'	删除1-3行

- "i" 在指定行前添加一个新行；"a" 在指定行后添加一个新行。

'3i\inserted'	在第3行前添加 "inserted"

- "c" 修改指定行：

'3c\changed'	将第3行改为 "changed"
'/number 1/c\hi'	将含有字 "number 1" 的行改为 "changed"。

- "y" 替换时应字符

'y/123/789/'	将所有 "1"/"2"/"3"，替换为 "7"/"8"/"9"。

- "p" 打印行。"=" 打印行号。"l" 打印不可打印的 ASCII 字符。
- "r" 在指定行后插入文件

'/number 1/r data1.txt'	在有 "number 1" 的行后插入所有 data1.txt 的内容。

- "-e" 选项允许在行多个命令

  sed -e 's/cat/dog/; s/xy/ab/' data.txt = sed 'S/cat/dog/; s/xy/ab/' data.txt

- "-f" 选项从文件中读取 sed 命令。
- "-n" 选项禁止输出其他行，仅打印包含匹配文本模式的行。
- 行寻址用来限制 sed 仅处理某些内容：

sed '2,3 s/dog/cat' data1.txt	对第2-3行替换
sed '/sus/s/dog/cat' data1.txt	对含有 "sus" 的行替换

- 可以用 ":" 等其他分隔符来代替 "/" 以避免转义。

- 正则表达式：
  - BRE：POSIX基础正则表达式
  - ERE：POSIX扩展正则表达式
- 特殊字符："*"，"["，"]"，"("，")"，"{"，"}"，"^"，"$"，"\"，"+"，"?"，"|"
  虽然"/"不是特殊字符，当作特殊字符一样使用其也需用"\"转义
- 锚字符："^"表示行首；"$"表示行尾
- 点号字符："."匹配除"\n"以外的任意一个字符
- 字符组："["和"]"中的任何一个字符都可以用来匹配；在第一个字母之前用"^"可以排除方括号中的字符用来匹配；可以用在破折号"-"来表示区间

  [0-5] = [012345]

  [a-cg-i] = [abcghi]

  - BRE特殊字符组：
    * [[:alpha:]] : [A-Za-z]
    * [[:alnum:]] : [0-9A-Za-z]
    * [[:blank:]] : 空格或tab
    * [[:digit:]] : [0-9]
    * [[:lower:]] : [a-z]
    * [[:print:]] : 可打印字符
    * [[:punct:]] : 标点
    * [[:space:]] : 空白字符：空格，"\t"，"NL"，"FF"，"VT"和"CR"
    * [[:upper:]] : [A-Z]

水平制表符 垂直制表符 一回车符

"\n" "\f" "\v" "\r"

- 星号字符："*"前面字符(组)出现0或多次

  b*t : b可以出现0次或无数次

  [ae]*t : a或e可以出现0次或无数次

以上是BRE，以下是ERE

- 问号字符："?"前面字符出现0或1次
- 加号字符："+"前面字符出现1次或多次
- 花括号："\{m\}"前面字符出现m次；"\{m,n\}"前面字符出现[m,n]次
- 管道符号："|"前后为逻辑"或"关系，即要么"|"前出现要么"|"后出现
- 分组："("和")"内的字符/字符串/表达式视为一个字符

· sed进阶：

- "n" 进入下一行

  '/header/{n;d}' 删除含有"header"行的下一行

- "N" 将此行和下一行当一行处理(但仍保留"\n")

  '/first/{N;s/\n/ /}' 与下一行合并.

  但如果没有下一行可供合并，会直接退出sed.

- "D" 删除模式空间中的第一行("\n"之前)

  '/^$/{N;/header/D}' 删除"header"前的空白行.

- "P" 打印模式空间中的第一行("\n"之前)

- 模式空间对某些行做处理时，可以用保存空间来保存一些行：

  * h：模式空间复制到保存空间
  * H：模式空间附加到保存空间
  * g：保存空间复制到模式空间
  * G：保存空间附加到模式空间
  * x：交换模式空间和保存空间.

- "!" 对符合匹配的行不执行命令，但对不符合匹配的所有行执行命令.

  sed -n '{1!G;h;$p}' data.txt 逆序打印文件中的每行.(也可以用 tac data.txt)

- "b" 为其前面的行执行分支命令，若未指明分支命令则跳过该行.

  sed '{/first/b jump1;s/This/No/;:jump1;s/This/Yes/}'

  对所有除"first"行替换成"No"；否则替换成"Yes".

  sed -n '{:start;s/.//p;/./b start}'

  将"."一个一个删除，但因为跳转到分支前面的标签上被形成了一个循环，每次删除一个逗号.

- "t" 根据命令的结果跳转到标签而非地址.

  sed '{s/first/matched/;t;s/This/No/}' data.txt

  将"first"换成"matched"，如果没有"first"，将"This"换成"No".

- "&" 符号用来替换命令中的匹配的模式. 可以用"("和")"定义替换模式中的多模式，在替代模式中使用特殊字符来引用每个独立的模式(以"\1"，"\2"，...表示). 在替换模式中使用圆括号时需用"\"转义以将其模式视为组字符.

  sed '{:start;s/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;t start}'

  可以用来在长数字中每三位间加一逗号.

· gawk

- 可以用 BEGIN/END 语段在处理代码前/后运行脚本，比如声明字段分隔符等。

```
gawk 'BEGIN{FS=","}{print $1,$2,$3}' data.txt
```

首先声明 data.txt 文件以","为分隔符，然后输出每行前3列。

- gawk 数据字段：
  - * FIELDWIDTHS：由空格分隔的一列数字，定义了每个数据字段确切的宽度。
  - * FS：输入字段分隔符
  - * OFS：输出字段分隔符
  - * RS：输入记录分隔符
  - * ORS：输出记录分隔符
- gawk 常用数据度量：
  - * FNR：当前文件数据行数
  - * NF：文件字段总数
  - * NR：已处理的输入记录数

如果只使用一个数据文件作为输入，FNR=NR；
.. 多个 .. .. ，FNR≤NR

引用 gawk 变量不加"$"。

- 可以在命令行上给 gawk 变量赋值，但此时变量在 BEGIN 部分不可用，所以需要用"-v"

```
<script> : BEGIN{FS=","}{print $n}
gawk -f script n=2 data.txt  => gawk -v n=2 -f script data.txt
```

- gawk 可以定义函数，但要出现在 BEGIN 前。
- gawk 匹配正则("~")和条件式("="/">"/"<")，出现在主代码块括号前。

· tar 命令可以用来归档：

```
tar -cf archive.tar /folder_to_be_archived 2>/dev/null
```

c 加入z以压缩为 tar.gz/tgz