1 查看文件夹大小

**du** –h –max-depth=1 .

2 查看系统运行时间

**uptime**

3**查看/proc/uptime文件计算系统运行时间**

**cat /proc/uptime**| awk -F. '{run\_days=$1 / 86400;run\_hour=($1 % 86400)/3600;run\_minute=($1 % 3600)/60;run\_second=$1 % 60;printf("系统已运行：%d天%d时%d分%d秒",run\_days,run\_hour,run\_minute,run\_second)}'

4**查看/proc/uptime文件计算系统启动时间**

date -d "$(awk -F. '{print $1}' /proc/uptime) second ago" +"%Y-%m-%d %H:%M:%S"

**5 awk**是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

**使用方法**

awk '{pattern + action}' {filenames}

尽管操作可能会很复杂，但语法总是这样，其中 **pattern** 表示 AWK 在数据中查找的内容，而 **action** 是在找到匹配内容时所执行的一系列命令。花括号（{}）不需要在程序中始终出现，但它们用于根据特定的模式对一系列指令进行分组。 pattern就是要表示的正则表达式，用斜杠括起来。

awk语言的最基本功能是在文件或者字符串中基于指定规则浏览和抽取信息，awk抽取信息后，才能进行其他文本操作。完整的awk脚本通常用来格式化文本文件中的信息。

*通常，awk是以文件的一行为处理单位的。awk每接收文件的一行，然后执行相应的命令*，来处理文本。

## ****调用awk****

有三种方式调用awk

***1.命令行方式***

awk [-F field-separator] 'commands' input-file(s)

其中，commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

***2.shell脚本方式***

将所有的awk命令插入一个文件，并使awk程序可执行，然后awk命令解释器作为脚本的首行，一遍通过键入脚本名称来调用。

相当于shell脚本首行的：#!/bin/sh

可以换成：#!/bin/awk

***3.将所有的awk命令插入一个单独文件，然后调用：***

awk -f awk-script-file input-file(s)

其中，-f选项加载awk-script-file中的awk脚本，input-file(s)跟上面的是一样的。

awk工作流程是这样的：读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域。默认域分隔符是"空白键" 或 "[tab]键

----------------------------------------

**awk是行处理器**: 相比较屏幕处理的优点，在处理庞大文件时不会出现内存溢出或是处理缓慢的问题，通常用来格式化文本信息

**awk处理过程:** 依次对每一行进行处理，然后输出

**awk命令形式:**

awk [-F|-f|-v] ‘BEGIN{} //{command1; command2} END{}’ file

 [-F|-f|-v]   大参数，-F指定分隔符，-f调用脚本，-v定义变量 var=value

'  '          引用代码块

BEGIN   初始化代码块，在对每一行进行处理之前，初始化代码，主要是引用全局变量，设置FS分隔符

//           匹配代码块，可以是字符串或正则表达式

{}           命令代码块，包含一条或多条命令

；          多条命令使用分号分隔

END      结尾代码块，在对每一行进行处理之后再执行的代码块，主要是进行最终计算或输出结尾摘要信息

**特殊要点:**

$0           表示整个当前行

$1           每行第一个字段

NF          字段数量变量

NR          每行的记录号，多文件记录递增

FNR        与NR类似，不过多文件记录不递增，每个文件都从1开始

\t            制表符

\n           换行符

FS          BEGIN时定义分隔符

RS       输入的记录分隔符， 默认为换行符(即文本是按一行一行输入)

~            匹配，与==相比不是精确比较

!~           不匹配，不精确比较

==         等于，必须全部相等，精确比较

!=           不等于，精确比较

&&　     逻辑与

||             逻辑或

+            匹配时表示1个或1个以上

/[0-9][0-9]+/   两个或两个以上数字

/[0-9][0-9]\*/    一个或一个以上数字

FILENAME 文件名

OFS      输出字段分隔符， 默认也是空格，可以改为制表符等

ORS        输出的记录分隔符，默认为换行符,即处理结果也是一行一行输出到屏幕

-F'[:#/]'   定义三个分隔符