awk, sed, grep 总结

在命令行其实只用过grep，其他两个用得很少，今天总结一下awk, sed, grep这linux三傻。

awk

1. awk是对文本进行格式化的工具，最擅长取列, 适合处理比较复杂的格式处理，主要的格式如下

```
awk [options] 'script' file1 file2, ...
```

```
awk [options] 'PATTERN {acticon}' file1 file2, ...
```

1. pattern部分决定动作语句何时触发及触发事件：BEGIN、END
2. action 对数据进行处理，放在{}内指明：print、printf
3. 最常用的是 print，默认以空白字符分隔

$0 代表整行，$1 代表第 1 段，$2 代表第 2 段，以此类推，$NF 代表最后一个字段，多个字段直接用逗号隔开

awk '{print $1, $2}' xxx.log

4. options 参数：输入分隔符，默认以空白字符分隔，通过 -F 选项来执行分隔符, 比如我有一个csv的文件分隔符是',' 那么我可以输出前两列

```
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge > awk -F ',' '{print $1,$2}' woman_ensemble_submission.csv
 player_a
0 ALBOT R.
1 ALBOT R.
2 ALBOT R.
3 ALBOT R.
4 ALBOT R.
5 ALBOT R.
6 ALBOT R.
7 ALBOT R.
```

5. 其他的一些参数

FS：输入字段分隔符，默认空白字符，一般需要加 -F

OFS：输出字段分隔符，默认是空格，一般需要加 -v

NF：分隔后的字段数量(意思就是有多少列)

NR：当前行的行号

比方下面这个例子，我的csv里面有5列，但是如果我给定NF==2, 那就没有。其他的判断条件还可以是其他的类似，如 NF>2，NF<4，$1==1234 等都是判断条件，很类似于基础版的dataframe列操作。

```
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge > awk -F ',' 'NF==2 {print $1,$2}' woman_ensemble_submission.csv
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge >                liangliang at Liangliangs-MacBook-Air.local (---)(main)
```

sed

sed 是流编辑器，是用来处理行数据的，处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”，接着用 sed 命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。

命令

```
sed [options]... 'script' inputfile
```

具体的选项:

1. -n：不输出模式空间内容到屏幕，即不自动打印,只输出匹配的行
2. -e：多点编辑
3. -f /PATH/SCRIPT_FILE：从指定文件中读取编辑脚本
4. -r：支持使用扩展正则表达式
5. -i：直接编辑文件
6. -i.bak：备份文件并原处编辑

script 地址定界

1. 不给地址：对全文进行处理
2. 单地址：

#：指定的行； $：最后一行

/pattern/：被此处模式所能够匹配到的每一行

编辑命令

1. d 删除模式空间匹配的行，并立即启用下一轮循环
2. p 显示符合条件的行，追加到默认输出之后
3. a [\]text1 在指定行后面追加文本,支持使用\n实现多行追加
4. i [\]text 在行前面插入文本
5. c [\]text 替换行为单行或多行文本
6. w /path/somefile 保存模式匹配的行至指定文件
7. r /path/somefile 读取指定文件的文本至模式空间中匹配到的行后
8. = 为模式空间中的行打印行号
9. ! 模式空间中匹配行取反处理
10. s///：查找替换,支持使用其它分隔符，s@@@，s###

替换标记：

1. g 行内全局替换
2. p 显示替换成功的行
3. w /PATH/TO/SOMEFILE 将替换成功的行保存至文件中

比如👇的🌰，就是输出第二行的内容。

```
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge > sed -n 2p woman_ensemble_submission.csv
0,ALBOT R.,ALTMAIER D.,0.0,1.0
```

或者需要打印1-5行：

```
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge > sed -n 1,5p woman_ensemble_submission.csv
,player_a,player_b,p(a_wins),p(b_wins)
0,ALBOT R.,ALTMAIER D.,0.0,1.0
1,ALBOT R.,ANDERSON K.,0.0,1.0
2,ALBOT R.,ANDUJAR P.,0.5276329183067672,0.4723670816932328
3,ALBOT R.,AUGER-ALIASSIME F.,0.0,1.0
```

sed进行文件替换并且输出 ，比方说我需要把ALBOT换成DJOKOVIC, 那应该用

```
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge ! > sed -i.be 's/ALBOT/DJOKOVIC/g' woman_ensemble_submission.csv
(dev_env_38) ~/Desktop/Euroclear_ML_Challenge > 
```

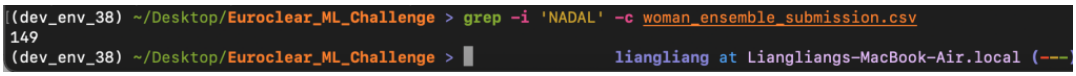
grep

- grep 强大的文本搜索工具，根据模式搜索文本，并将符合模式的文本行显示出来。
- gerp 命令格式：

```
grep 【option】 pattern 【file】
```

option：

- -i：忽略字符大小写
- -n：显示匹配的行号
- -v：显示没有被匹配的行
- –color：将匹配的字符以高亮颜色标记出来
- -c：统计匹配的行数
- -o：仅显示匹配到的字符串
- -q：静默模式，不输出任何信息
- -e：实现多个选项间的逻辑 or 关系
- -v：反转查找
- -w：匹配整个单词
- -A：after ，显示后行
- -B：before ，显示前行
- -C：context，显示前后行
- -E：相当于 egrep，即 grep -E = egrep



发布者

LiangLiang ZHENG

Data Scientist based in Belgium, passionate about Data Analysis/Deep Learning.

查看LiangLiang ZHENG的所有文章 →

2021-06-17 LiangLiang ZHENG Linux & CLI

在WordPress.com的博客.