

天津医科大学理论课教案首页

(共 6 页、第 1 页)

| | |
|---------------------------------|-------------------------------|
| 课程名称：Linux 系统概论 | 课程内容/章节：高级 Linux 命令 / 第 8&9 章 |
| 教师姓名：伊现富 | 职称：讲师 |
| 教学日期：2018 年 6 月 4 日 10:00-12:00 | |
| 授课对象：生物医学工程与技术学院 2016 级生信班（本） | 听课人数：28 |
| 授课方式：理论讲授 | 学时数：2 |
| 教材版本：Unix 入门经典，第 1 版 | |

教学目的与要求（分掌握、熟悉、了解、自学四个层次）：

- 掌握常见的元字符，find 的使用，grep 的使用，sed 的使用，AWK 的使用。
- 熟悉正则表达式的构建，find 的常用测试，grep 的常用选项，sed 的工作原理，AWK 的工作原理和脚本结构。
- 了解 Linux 中的文本处理命令。
- 自学文本处理命令在生物信息学中的应用。

授课内容及学时分配：

- (5') 引言与导入：回顾 Linux 命令的基本格式以及常用的 Linux 命令，引申出授课内容。
- (20') 正则表达式和元字符：介绍正则表达式和元字符的概念，讲解元字符中的字符、字符集、量词和边界，举例说明正则表达式的构建。
- (20') find 和 grep：讲解 find 的常用测试并通过实例进行演示，讲解 grep 的常用选项并通过实例进行演示。
- (5') 文本处理命令：总结 Linux 中常用的文本处理命令。
- (40') sed 和 AWK：讲解 sed 的工作原理、定位规则和常用命令，通过实例演示 sed 的使用；讲解 AWK 的工作原理和脚本结构，通过实例演示 AWK 的使用。
- (5') 生物信息学中的应用：举例说明文本处理命令在生物信息学中的应用。
- (5') 总结与答疑：总结授课内容中的知识点与技能，解答学生疑问。

教学重点、难点及解决策略：

- 重点：元字符，find 的使用，grep 的使用。
- 难点：元字符，sed 的使用，AWK 的使用。
- 解决策略：通过实例讲解与操作演示帮助学生理解、记忆。

专业外语词汇或术语：

| | |
|----------------------------|-----------------------|
| 正则表达式 (regular expression) | 字符集 (character class) |
| 元字符 (metacharacter) | 模式空间 (pattern space) |

辅助教学情况：

- 多媒体：元字符，find 的测试，grep 的选项，sed 和 AWK 的工作原理。
- 板书：常见的元字符，正则表达式的构建与解析。
- 演示：find、grep、sed 和 AWK 的基本使用。

复习思考题：

- | | |
|----------------------|------------------------|
| • 列举五个常见的元字符并解释其含义。 | • 根据要求组合使用文本处理命令。 |
| • 根据要求编写正则表达式。 | • 根据要求编写 sed 脚本编辑文件。 |
| • 根据要求使用 find 查找文件。 | • 根据要求编写 AWK 脚本输出特定字段。 |
| • 根据要求使用 grep 查找字符串。 | • 根据要求处理生物信息学文本数据。 |

参考资料：

- (美) Harley Hahn 著，张杰良 译。Unix & Linux 大学教程，清华大学出版社，2010。
- 鸟哥 著，王世江 改编。鸟哥的 Linux 私房菜——基础学习篇（第三版），人民邮电出版社，2010。
- (美) Ben Forta 著，杨涛 等译。正则表达式必知必会，人民邮电出版社，2007。
- 维基百科等网络资源。

主任签字：

年 月 日

教务处制

一、引言与导入 (5 分钟)

1. 回顾: Linux 命令的基本格式, 常用的 Linux 命令
2. 介绍: 正则表达式和元字符, find、grep、sed、AWK 等高级命令

二、正则表达式和元字符 (20 分钟)

1. 简介

- 正则表达式: 具有一定句法的集合或短语, 包含元字符或普通字符, 表示某类文本或字符串。
 - 元字符: 代表一组字符或命令的字符, 用最小的字符集表示多组文本。
 - 相关命令: less, more, grep, sed, AWK, vim, ……
2. **【重点、难点】** 元字符 (结合实例讲解每一个元字符)
 - 字符: 一般字符, ., [], [a-z], [0-9], [^], \
 - 字符集: \d, \D, \s, \S, w, \W
 - 量词: ?, *, +, {m}, {m,n}
 - 边界: ^, \$
 - 其他: (), |

| 正则表达式 | 描 述 | 示 例 |
|-------|-----------------------|---|
| ^ | 行起始标记 | ^tux 匹配以tux起始的行 |
| \$ | 行尾标记 | tux\$ 匹配以tux结尾的行 |
| . | 匹配任意一个字符 | Hack. 匹配Hackl和Hacki, 但是不能匹配Hackl2和Hackil, 它只能匹配单个字符 |
| [] | 匹配包含在[字符]之中的任意一个字符 | coo[kl] 匹配cook或cool |
| [^] | 匹配除[^字符]之外的任意一个字符 | 9[^01] 匹配92、93, 但是不匹配91或90 |
| [-] | 匹配[]中指定范围内的任意一个字符 | [1-5] 匹配从1~5的任意一个数字 |
| ? | 匹配之前的项1次或0次 | colou?r 匹配color或colour, 但是不能匹配colourr |
| + | 匹配之前的项1次或多次 | Rollno-9+ 匹配Rollno-99、Rollno-9, 但是不能匹配Rollno- |
| * | 匹配之前的项0次或多次 | co*l 匹配cl、col、cool等 |
| () | 创建一个用于匹配的子串 | ma(tri)? 匹配max或maxtrix |
| {n} | 匹配之前的项n次 | [0-9]{3} 匹配任意一个三位数, [0-9]{3}可以扩展为[0-9][0-9][0-9] |
| {n,} | 之前的项至少需要匹配n次 | [0-9]{2,} 匹配任意一个两位或更多位的数字 |
| {n,m} | 指定之前的项所必需匹配的最小次数和最大次数 | [0-9]{2,5} 匹配从两位数到五位数之间的任意一个数字 |
| | 交替——匹配 两边的任意一项 | Oct (1st 2nd) 匹配Oct 1st或Oct 2nd |
| \ | 转义符可以将上面介绍的特殊字符进行转义 | a\.b 匹配a.b, 但不能匹配ajb。通过在 . 之间加上前缀 \, 从而忽略了 . 的特殊意义 |

3. 正则表达式

a. /^[A-Z]..\$/ 查找文本中所有以大写字母开头、后跟两个任意字符, 再跟一个换行符的行。查找结果是第 5 行的 Dan。

b. /^[A-Z][a-z]*3[0-5]/ 查找所有以大写字母开头、后跟零个或多个小写字母或空格, 再跟数字 3 和一个 0~5 之间的数字的行。查找结果是第 2 行。

c. /[a-z]*\./ 查找包含跟在零个或多个小写字母后的句点的行。结果是第 1、2、7、8 行。

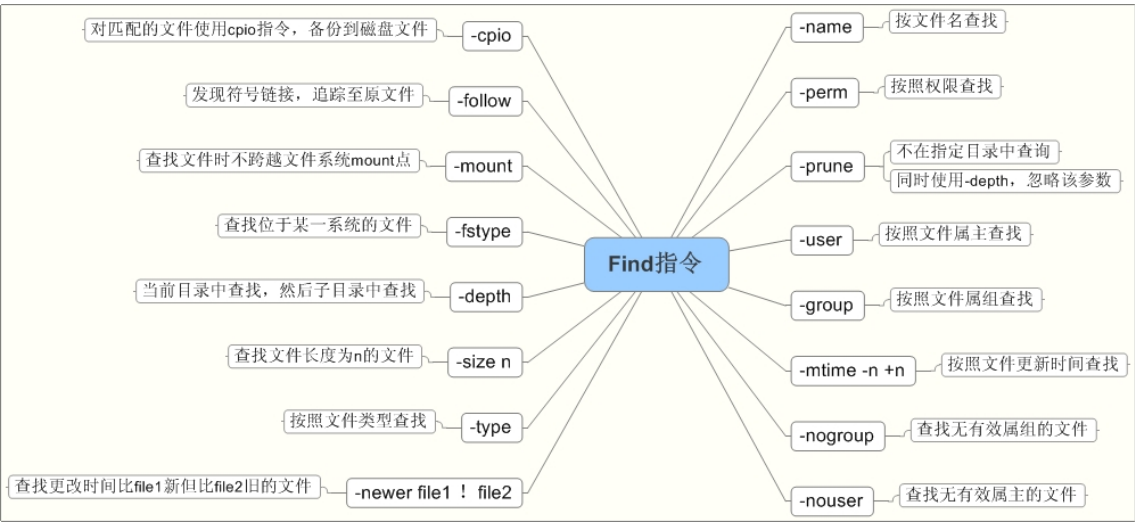
d. /^ *[A-Z][a-z][a-z]\$/ 查找以零个或多个空格开头(注意: 制表符不算空格), 后跟一个大写字母、两个小写字母和一个换行符的行。结果将是第 4 行的 Tom 和第 5 行的 Dan。

e. /^[A-Za-z]*^[A-Za-z]*\$/ 查找以零个或多个大/小写字母开头, 后跟一个非逗号的字符, 再跟零个或多个大/小写字母和一个换行符的行。结果是第 5 行。

三、 grep 和 find (20 分钟)

1. 【重点】 find (实例讲解、操作演示)

(1) 测试



(2) 实例

- find /etc -name passwd
- find /home -user USER
- find /home -size +2G -print

```
find . -name "*.jpg" -exec rm -fv {} \;
```

找出以".jpg"结尾的文件

rm -fv 进行删除操作

{} 代表find过滤出来的以".jpg"结尾的文件

": -exec参数指定的command结束符; "\"对后面的分号进行转义"

2. 【重点】 grep (实例讲解、操作演示)

(1) 简介

定义 Globally search a Regular Expression and Print (全局正则表达式打印)

功能 在文件中搜索用户所指定的序列, 然后将结果打印出来

用途 搜索文件或者在文件中进行查找

结构 grep StringToSearchFor FileToSearch

(2) 选项

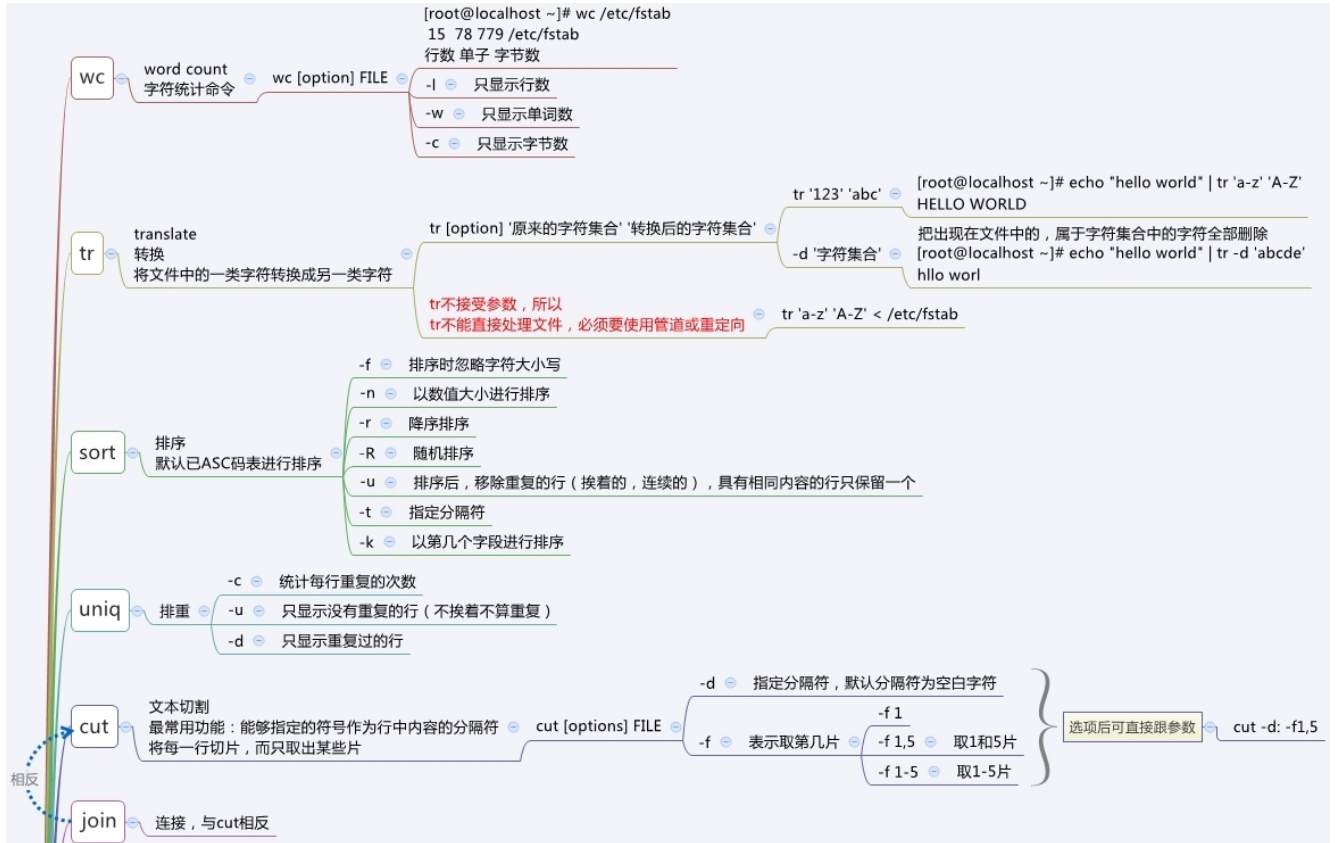
grep [options] 'Pattern' FILE

- color=auto: 为匹配的项自动添加颜色
- v: 反向选取, 只显示不符合模式的行。
- o: 只显示被模式匹配到的字符串, 而不是整个行
- i: 忽略大小写
- A n: 显示匹配到的行时, 顺带显示其后面的n行, A表示after
- B n: 前面的n行
- C n: 前后的n行
- E: 等于egrep, 表示使用扩展的正则表达式
- w: 字符正则表达式, 类似于锚定行首
- r: 递归搜索

(3) 实例

| 命令 | 显示 |
|---------------------------|---------------------------------|
| grep '[A-Z]' list | list 中包含一个大写字母的行 |
| grep '[0-9]' data | data 中包含数字的行 |
| grep '[A-Z]...[0-9]' list | list 中包含以大写字母开始、数字结尾的 5 个字符组合的行 |
| grep '\.pic\$' filelist | filelist 中以.pic 结尾的行 |

四、 文本处理命令 (5 分钟)



cat ~/.bash_history | cut -d " " -f1 | sort | uniq -c | sort -nr | head

delimiter 用空格分隔出的第一部分 count number+reverse

为了使uniq起作用，所有的重复行必须是相邻的

| 命令 | 说明 | 命令 | 说明 | 命令 | 说明 |
|-------|------------|-------|-------------|--------|--------------|
| cat | 组合文件 | split | 分割文件 | join | 合并两个文件中的有序数据 |
| tac | 反转文本行的顺序 | rev | 反转字符串的顺序 | tr | 转换字符 |
| head | 从数据开头选择数据行 | tail | 从数据末尾选择数据行 | comm | 比较有序文本文件 |
| less | 显示文件 | more | 显示文件 | expand | 将制表符转换成空格 |
| colrm | 删除数据列 | cmp | 比较任意两个文件 | tee | 管道分流 |
| cut | 抽取数据列/字段 | paste | 组合数据列 | grep | 选取包含特定模式的行 |
| nl | 创建行号 | wc | 统计行/单词/字符数量 | sed | 非交互式文本编辑 |
| fold | 格式化行 | fmt | 格式化段落 | shuf | 随机选取行/文件 |
| sort | 排序数据 | uniq | 查找重复行 | | |

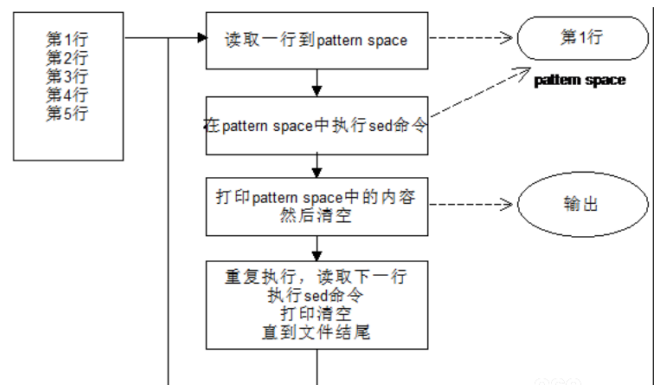
五、 sed 和 AWK (40 分钟)

1. 简介

- sed: 处理纯文本流的文本编辑器
- AWK: 一种输出格式化语言
- 对象: 已有文本 (管道, 命令, 文本文件)

2. 【难点】sed (实例讲解、操作演示)

- (1) 工作原理
- (2) 定位
- (3) 命令



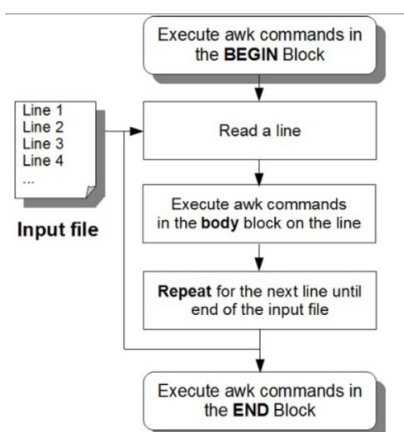
(4) 实例

- `sed -n '1,3!p' 123.txt`
- `sed -n -e '/the/p' -e '/the/= ' 123.txt`
- `sed 's/this/that/g' 123.txt`
- `echo "hello" | sed 's/$/.txt/g'`
- `ls -l | sed -n '1,3p'`

| 命令 | 助记 | 说明 | 命令 | 说明 |
|----|------------|-------------|-----------------------|----------------------|
| p | Print | 打印匹配行 | n | 第 n 行 |
| = | — | 显示匹配行的行号 | \$ | 最后一行 |
| d | Delete | 删除匹配行 | m,n | 从第 m 行到第 n 行 |
| a\ | Append | 在指定行后面追加文本 | /pattern/ | 包含指定模式的行 |
| i\ | Insert | 在指定行后面插入文本 | /pattern/,n | 从包含指定模式的行到第 n 行 |
| c\ | Correct | 用新文本替换指定行 | n,/pattern/ | 从第 n 行到包含指定模式的行 |
| s | Substitute | 替换命令 | /pattern1/,/pattern2/ | 从包含模式 1 的行到包含模式 2 的行 |
| l | List | 显示指定行中的所有字符 | ! | 反向选择, 不包含指定行, 如 m,n! |
| r | Read | 读取文件 | | |
| w | Write | 写入文件 | | |
| n | Next | 读取指定行的下一行 | | |
| q | Quit | 退出 sed | | |

3. 【难点】AWK (实例讲解、操作演示)

(1) 工作原理



- ① 如果存在 **BEGIN** , **awk** 首先执行它指定的 **actions**
- ② **awk** 从输入中读取一行, 称为一条输入记录
- ③ **awk** 将读入的记录分割成数个字段, 并将第一个字段放入变量 **\$1** 中, 第二个放入变量 **\$2** 中, 以此类推; **\$0** 表示整条记录; 字段分隔符可以通过选项 **-F** 指定, 否则使用缺省的分隔符。
- ④ 把当前输入记录依次与每一个 **awk_cmd** 中 **pattern** 比较: 如果相匹配, 就执行对应的 **actions**; 如果不匹配, 就跳过对应的 **actions**, 直到完成所有的 **awk_cmd**
- ⑤ 当一条输入记录处理完毕后, **awk** 读取输入的下一行, 重复上面的处理过程, 直到所有输入全部处理完毕。
- ⑥ **awk** 处理完所有的输入后, 若存在 **END**, 执行相应的 **actions**
- ⑦ 如果输入是文件列表, **awk** 将按顺序处理列表中的每个文件。

(2) 脚本结构

```

1 awk
2
3 '
4 BEGIN {actions}
5
6 /pattern1/{actions}
7 .....
8 /patternN/{actions}
9
10 END {actions}
11 '
12
13 InputFile
    
```

| 记录 | 字段 1 (名字) | 字段 2 (姓) | 字段 3 (报酬率) | 字段 4 (小时数) |
|-------|-----------|------------|------------|------------|
| 记录 2 | Susan | White | 6.00 | 23 |
| | Mark | Eagle | 6.25 | 40 |
| 记录 4 | Tuan | Nguyen | 7.89 | 44 |
| | Dan | Black | 7.23 | 40 |
| | Amanda | Trapp | 6.95 | 40 |
| | Brian | Devaux | 7.95 | 0 |
| | Chris | Walljasper | 6.89 | 32 |
| | Mary | Lamb | 8.22 | 40 |
| 记录 10 | Jackie | Kammaoto | 7.59 | 40 |
| | Nicky | Barber | 6.35 | 40 |

有 10 个记录的文件, 每个记录 4 个字段

(3) 常见变量

(4) 记录与字段

(5) 命令构成

- 编辑命令 (一个、一组命令或一个命令文件)
 - 模式: 只编辑与模式相匹配的记录行; 没有提供模式时, 匹配所有行
 - 命令: 具体执行的编辑命令; 没有指定命令时, 打印整个记录行
- 要编辑的数据 (数据或数据文件)

(6) 实例

- `awk '{print $0}' 123.txt`
- `ls -l | awk '{if($1 !~ /^d/) {print $0}}'`
- `awk '{printf("%03d %s\n",NR,$0)}' ori.txt > dst.txt`
- `awk 'BEGIN{FS=" ";OFS="\t"}{print $1,$2} ori.txt > dst.txt`

| 变量 | 描述 |
|-------------|----------------------|
| \$n | 当前记录的第n个字段，字段间由FS分隔。 |
| \$0 | 完整的输入记录。 |
| ARGC | 命令行参数的数目。 |
| ARGIND | 命令行中当前文件的位置(从0开始算)。 |
| ARGV | 包含命令行参数的数组。 |
| CONVFMT | 数字转换格式(默认值为%.6g) |
| ENVIRON | 环境变量关联数组。 |
| ERRNO | 最后一个系统错误的描述。 |
| FIELDWIDTHS | 字段宽度列表(用空格键分隔)。 |
| FILENAME | 当前文件名。 |

| | |
|------------|------------------------|
| FNR | 同NR，但相对于当前文件。 |
| FS | 字段分隔符(默认是任何空格)。 |
| IGNORECASE | 如果为真，则进行忽略大小写的匹配。 |
| NF | 当前记录中的字段数。 |
| NR | 当前记录数。 |
| OFMT | 数字的输出格式(默认值是%.6g)。 |
| OFS | 输出字段分隔符(默认值是一个空格)。 |
| ORS | 输出记录分隔符(默认值是一个换行符)。 |
| RLENGTH | 由match函数所匹配的字符串的长度。 |
| RS | 记录分隔符(默认是一个换行符)。 |
| RSTART | 由match函数所匹配的字符串的第一个位置。 |
| SUBSEP | 数组下标分隔符(默认值是\034)。 |

六、 生物信息学中的应用 (5 分钟)

1. 实例 (对命令进行逐一解析)

- `grep '>' file.fasta | wc -l`
- `awk '/^>/{s=++d".fa"} {print > s}' multi.fa`
- `sed -n '1~4s/^@/>/p;2~4p' file.fq > file.fa`
- `cat file.txt | awk '$1=="1"' |
awk '$3>=1000000' | awk '$3<=2000000'`
- `echo {A,C,T,G}{A,C,T,G}{A,C,T,G}`
- `grep -c '$\tgene\t' yourannots.gff3`
- `grep -v '^#' GFF3 | cut -s -f 3 | sort | uniq`
- `cut -f1,6 gene.bed | sort | uniq -c | sort -nr`
- `cat myfile.fq | awk '((NR-2)%4==0){read=$1;total++;count[read]++;
END{for(read in count){if(!max||count[read]>max)
{max=count[read];maxRead=read};if(count[read]==1){unique++}};
print total,unique,unique*100/total,maxRead,count[maxRead],
count[maxRead]*100/total}'`

七、 总结与答疑 (5 分钟)

1. 知识点

- 元字符：字符，字符集，量词，边界
- 正则表达式：构建，解析
- `find`：常用测试，日常使用
- `grep`：常用选项，日常使用
- 文本处理命令：`cut`, `wc`, `sort`, `uniq`, ...
- `sed`：工作原理，定位方式，编辑命令
- `AWK`：脚本结构，工作原理，常见变量，记录，字段

2. 技能

- 解析并编写正则表达式
- `find` 和 `grep` 的基本用法
- 文本处理命令的日常应用
- 使用 `sed` 和 `AWK` 处理文本
- 正则表达式在 `grep`、`sed` 和 `AWK` 中的应用