文本间隔：

```
# 在每一行后面增加一空行

sed G
awk '{printf("%s\n\n",$0)}'

# 将原来的所有空行删除并在每一行后面增加一空行。
# 这样在输出的文本中每一行后面将有且只有一空行。

sed '/^$/d;G'
awk '!/^$/{printf("%s\n\n",$0)}'

# 在每一行后面增加两行空行

sed 'G;G'
awk '{printf("%s\n\n\n",$0)}'

# 将第一个脚本所产生的所有空行删除（即删除所有偶数行）

sed 'n;d'
awk '{f=!f;if(f)print $0}'

# 在匹配式样“regex”的行之前插入一空行

sed '/regex/{x;p;x;}'
awk '{if(/regex/)printf("\n%s\n",$0);else print $0}'

# 在匹配式样“regex”的行之后插入一空行

sed '/regex/G'
awk '{if(/regex/)printf("%s\n\n",$0);else print $0}'

# 在匹配式样“regex”的行之前和之后各插入一空行

sed '/regex/{x;p;x;G;}'
awk '{if(/regex/)printf("\n%s\n\n",$0);else print $0}'
```

编号：

```
# 为文件中的每一行进行编号（简单的左对齐方式）。这里使用了“制表符”
# （tab，见本文末尾关于’\t’的用法的描述）而不是空格来对齐边缘。

sed = filename | sed 'N;s/\n/\t/'
awk '{i++;printf("%d\t%s\n",i,$0)}'
```

```
# 对文件中的所有行编号（行号在左，文字右端对齐）。

sed = filename | sed 'N; s/^/         /; s/ *\(.\{6,\}\)\n/\1    /'
awk '{i++;printf("%6d    %s\n",i,$0)}'

# 对文件中的所有行编号，但只显示非空白行的行号。

sed '/./=' filename | sed '/./N; s/\n/ /'
awk '{i++;if(!/^$/)printf("%d %s\n",i,$0);else print}'

# 计算行数 （模拟 “wc -l”）

sed -n '$='
awk '{i++}END{print i}'
```

文本转换和替代：

```
# Unix 环境：转换 DOS 的新行符（CR/LF）为 Unix 格式。

sed 's/.$//'                                # 假设所有行以 CR/LF 结束
sed 's/^M$//'                               # 在 bash/tcsh 中，将按 Ctrl-M 改为按 Ctrl-V
sed 's/\x0D$//'                             # ssed、gsed 3.02.80，及更高版本
awk '{sub(/\x0D$/,"");print $0}'

# Unix 环境：转换 Unix 的新行符（LF）为 DOS 格式。

sed "s/$/`echo -e \\\r`/"                   # 在 ksh 下所使用的命令
sed 's/$"/`echo \\\r`/"                     # 在 bash 下所使用的命令
sed "s/$/`echo \\\r`/"                      # 在 zsh 下所使用的命令
sed 's/$/\r/'                               # gsed 3.02.80 及更高版本
awk '{printf("%s\r\n",$0)}'

# DOS 环境：转换 Unix 新行符（LF）为 DOS 格式。

sed "s/$//"                                 # 方法 1
sed -n p                                    # 方法 2

DOS 环境的略过

# DOS 环境：转换 DOS 新行符（CR/LF）为 Unix 格式。
# 下面的脚本只对 UnxUtils sed 4.0.7 及更高版本有效。要识别 UnxUtils 版本的
# sed 可以通过其特有的“-text”选项。你可以使用帮助选项（“-help”）看
# 其中有无一个“-text”项以此来判断所使用的是否是 UnxUtils 版本。其它 DOS
# 版本的的 sed 则无法进行这一转换。但可以用“tr”来实现这一转换。

sed "s/\r//" infile >outfile        # UnxUtils sed v4.0.7 或更高版本
```

```
tr -d \r <infile >outfile            # GNU tr 1.22 或更高版本
```

DOS 环境的略过

```
# 将每一行前导的“空白字符”（空格，制表符）删除
# 使之左对齐

sed 's/^[ \t]*//'                    # 见本文末尾关于'\t'用法的描述
awk '{sub(/^[ \t]+/,"");print $0}'

# 将每一行拖尾的“空白字符”（空格，制表符）删除

sed 's/[ \t]*$//'                    # 见本文末尾关于'\t'用法的描述
awk '{sub(/[ \t]+$/,"");print $0}'

# 将每一行中的前导和拖尾的空白字符删除

sed 's/^[ \t]*//;s/[ \t]*$//'
awk '{sub(/^[ \t]+/,"");sub(/[ \t]+$/,"");print $0}'

# 在每一行开头处插入 5 个空格（使全文向右移动 5 个字符的位置）

sed 's/^/     /'
awk '{printf("     %s\n",$0)}'

# 以 79 个字符为宽度，将所有文本右对齐
# 78 个字符外加最后的一个空格

sed -e :a -e 's/^.\{1,78\}$/ &/;ta'
awk '{printf("%79s\n",$0)}'

# 以 79 个字符为宽度，使所有文本居中。在方法 1 中，为了让文本居中每一行的前
# 头和后头都填充了空格。 在方法 2 中，在居中文本的过程中只在文本的前面填充
# 空格，并且最终这些空格将有一半会被删除。此外每一行的后头并未填充空格。

sed    -e :a -e 's/^.\{1,77\}$/ & /;ta'                    # 方法 1
sed    -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/\( *\)\1/\1/'  # 方法 2
awk '{for(i=0;i<39-length($0)/2;i++)printf(" ");printf("%s\n",$0)}'  #相当于上面的方法二

# 在每一行中查找字串“foo”，并将找到的“foo”替换为“bar”

sed 's/foo/bar/'                     # 只替换每一行中的第一个“foo”字串
sed 's/foo/bar/4'                    # 只替换每一行中的第四个“foo”字串
sed 's/foo/bar/g'                    # 将每一行中的所有“foo”都换成“bar”
sed 's/\(.*\)foo\(.*foo\)/\1bar\2/' # 替换倒数第二个“foo”
sed 's/\(.*\)foo/\1bar/'             # 替换最后一个“foo”
```

```
awk '{gsub(/foo/,"bar");print $0}'    # 将每一行中的所有“foo”都换成“bar”

# 只在行中出现字串“baz”的情况下将“foo”替换成“bar”

sed '/baz/s/foo/bar/g'
awk '{if(/baz/)gsub(/foo/,"bar");print $0}'

# 将“foo”替换成“bar”，并且只在行中未出现字串“baz”的情况下替换

sed '/baz/!s/foo/bar/g'
awk '{if(/baz$/)gsub(/foo/,"bar");print $0}'

# 不管是“scarlet”“ruby”还是“puce”，一律换成“red”

sed 's/scarlet/red/g;s/ruby/red/g;s/puce/red/g'   #对多数的 sed 都有效
gsed 's/scarlet\|ruby\|puce/red/g'                # 只对 GNU sed 有效
awk '{gsub(/scarlet|ruby|puce/,"red");print $0}'

# 倒置所有行，第一行成为最后一行，依次类推（模拟“tac”）。
# 由于某些原因，使用下面命令时 HHsed v1.5 会将文件中的空行删除

sed '1!G;h;$!d'                   # 方法 1
sed -n '1!G;h;$p'                 # 方法 2
awk '{A[i++]=$0}END{for(j=i-1;j>=0;j--)print A[j]}'

# 将行中的字符逆序排列，第一个字成为最后一字，……（模拟“rev”）

sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'
awk '{for(i=length($0);i>0;i--)printf("%s",substr($0,i,1));printf("\n")}'

# 将每两行连接成一行（类似“paste”）

sed '$!N;s/\n/ /'
awk '{f=!f;if(f)printf("%s",$0);else printf(" %s\n",$0)}'

# 如果当前行以反斜杠“\”结束，则将下一行并到当前行末尾
# 并去掉原来行尾的反斜杠

sed -e :a -e '/\\$/N; s/\\\n//; ta'
awk '{if(/\\$/)printf("%s",substr($0,0,length($0)-1));else printf("%s\n",$0)}'

# 如果当前行以等号开头，将当前行并到上一行末尾
# 并以单个空格代替原来行头的“=”

sed -e :a -e '$!N;s/\n=/ /;ta' -e 'P;D'
awk '{if(/^=/)printf(" %s",substr($0,2));else printf("%s%s",a,$0);a="\n"}END{printf("\n")}'
```

为数字字串增加逗号分隔符号，将“1234567”改为“1,234,567”

```
gsed ':a;s/\B[0-9]\{3\}\>/,&/;ta'                    # GNU sed
sed -e :a -e 's/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta'   # 其他 sed
```

#awk 的正则没有后向匹配和引用，搞的比较狼狈，呵呵。

```
awk
'{while(match($0,/[0-9][0-9][0-9][0-9]+/)){$0=sprintf("%s,%s",substr($0,0,RSTART+RLENGTH-4),substr($0,RSTART+RLENGTH-3))}print $0}'
```

为带有小数点和负号的数值增加逗号分隔符（GNU sed）

```
gsed -r ':a;s/(^|[^0-9.])([0-9]+)([0-9]{3})/\1\2,\3/g;ta'
```

#和上例差不多

```
awk
'{while(match($0,/[^\.0-9][0-9][0-9][0-9][0-9]+/)){$0=sprintf("%s,%s",substr($0,0,RSTART+RLENGTH-4),substr($0,RSTART+RLENGTH-3))}print $0}'
```

在每 5 行后增加一空白行 （在第 5，10，15，20，等行后增加一空白行）

```
gsed '0~5G'                          # 只对 GNU sed 有效
sed 'n;n;n;n;G;'                     # 其他 sed
awk '{print $0;i++;if(i==5){printf("\n");i=0}}'
```

选择性地显示特定行：

显示文件中的前 10 行 （模拟“head”的行为）

```
sed 10q
awk '{print;if(NR==10)exit}'
```

显示文件中的第一行 （模拟“head -1”命令）

```
sed q
awk '{print;exit}'
```

显示文件中的最后 10 行 （模拟“tail”）

```
sed -e :a -e '$q;N;11,$D;ba'
```

#用 awk 干这个有点亏，得全文缓存，对于大文件肯定很慢

```
awk '{A[NR]=$0}END{for(i=NR-9;i<=NR;i++)print A[i]}'
```

显示文件中的最后 2 行（模拟“tail -2”命令）

```
sed '$!N;$!D'
awk '{A[NR]=$0}END{for(i=NR-1;i<=NR;i++)print A[i]}'
```

显示文件中的最后一行（模拟“tail -1”）

```
sed '$!d'                              # 方法 1
sed -n '$p'                            # 方法 2
```

#这个比较好办，只存最后一行了。

```
awk '{A=$0}END{print A}'
```

显示文件中的倒数第二行

```
sed -e '$!{h;d;}' -e x                 # 当文件中只有一行时，输出空行
sed -e '1{$q;}' -e '$!{h;d;}' -e x     # 当文件中只有一行时，显示该行
sed -e '1{$d;}' -e '$!{h;d;}' -e x     # 当文件中只有一行时，不输出
```

#存两行呗（当文件中只有一行时，输出空行）

```
awk '{B=A;A=$0}END{print B}'
```

只显示匹配正则表达式的行（模拟“grep”）

```
sed -n '/regexp/p'                     # 方法 1
sed '/regexp/!d'                       # 方法 2
awk '/regexp/{print}'
```

只显示“不”匹配正则表达式的行（模拟“grep -v”）

```
sed -n '/regexp/!p'                    # 方法 1，与前面的命令相对应
sed '/regexp/d'                        # 方法 2，类似的语法
awk '!/regexp/{print}'
```

查找“regexp”并将匹配行的上一行显示出来，但并不显示匹配行

```
sed -n '/regexp/{g;1!p;};h'
awk '/regexp/{print A}{A=$0}'
```

查找“regexp”并将匹配行的下一行显示出来，但并不显示匹配行

```
sed -n '/regexp/{n;p;}'
awk '{if(A)print;A=0}/regexp/{A=1}'
```

显示包含“regexp”的行及其前后行，并在第一行之前加上“regexp”所在行的行号（类似“grep -A1 -B1”）

```
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h
awk '{if(F)print;F=0}/regexp/{print NR;print b;print;F=1}{b=$0}'
```

显示包含“AAA”、“BBB”和“CCC”的行（任意次序）

```
sed '/AAA/!d; /BBB/!d; /CCC/!d'    # 字串的次序不影响结果
awk '{if(match($0,/AAA/) && match($0,/BBB/) && match($0,/CCC/))print}'
```

显示包含“AAA”、“BBB”和“CCC”的行（固定次序）

```
sed '/AAA.*BBB.*CCC/!d'
awk '{if(match($0,/AAA.*BBB.*CCC/))print}'
```

显示包含“AAA”“BBB”或“CCC”的行 （模拟“egrep”）

```
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d      # 多数 sed
gsed '/AAA\|BBB\|CCC/!d'                          # 对 GNU sed 有效
awk '/AAA/{print;next}/BBB/{print;next}/CCC/{print}'
awk '/AAA|BBB|CCC/{print}'
```

显示包含“AAA”的段落 （段落间以空行分隔）
HHsed v1.5 必须在“x;”后加入“G;”，接下来的 3 个脚本都是这样

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'
awk 'BEGIN{RS=""}/AAA/{print}'
awk -vRS= '/AAA/{print}'
```

显示包含“AAA”“BBB”和“CCC”三个字串的段落 （任意次序）

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;/BBB/!d;/CCC/!d'
awk -vRS= '{if(match($0,/AAA/) && match($0,/BBB/) && match($0,/CCC/))print}'
```

显示包含“AAA”、“BBB”、“CCC”三者中任一字串的段落 （任意次序）

```
sed -e '/./{H;$!d;}' -e 'x;/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
gsed '/./{H;$!d;};x;/AAA\|BBB\|CCC/b;d'           # 只对 GNU sed 有效
awk -vRS= '/AAA|BBB|CCC/{print "";print}'
```

显示包含 65 个或以上字符的行

```
sed -n '/^.\{65\}/p'
cat ll.txt | awk '{if(length($0)>=65)print}'
```

```
# 显示包含 65 个以下字符的行

sed -n '/^.\{65\}/!p'                    # 方法 1，与上面的脚本相对应
sed '/^.\{65\}/d'                        # 方法 2，更简便一点的方法
awk '{if(length($0)<=65)print}'

# 显示部分文本——从包含正则表达式的行开始到最后一行结束

sed -n '/regexp/,$p'
awk '/regexp/{F=1}{if(F)print}'

# 显示部分文本——指定行号范围（从第 8 至第 12 行，含 8 和 12 行）

sed -n '8,12p'                           # 方法 1
sed '8,12!d'                             # 方法 2
awk '{if(NR>=8 && NR<12)print}'

# 显示第 52 行

sed -n '52p'                             # 方法 1
sed '52!d'                               # 方法 2
sed '52q;d'                              # 方法 3, 处理大文件时更有效率
awk '{if(NR==52){print;exit}}'

# 从第 3 行开始，每 7 行显示一次

gsed -n '3~7p'                           # 只对 GNU sed 有效
sed -n '3,${p;n;n;n;n;n;n;}'         # 其他 sed
awk '{if(NR==3)F=1}{if(F){i++;if(i%7==1)print}}'

# 显示两个正则表达式之间的文本（包含）

sed -n '/Iowa/,/Montana/p'           # 区分大小写方式
awk '/Iowa/{F=1}{if(F)print}/Montana/{F=0}'
```

选择性地删除特定行：

```
# 显示通篇文档，除了两个正则表达式之间的内容

sed '/Iowa/,/Montana/d'
awk '/Iowa/{F=1}{if(!F)print}/Montana/{F=0}'

# 删除文件中相邻的重复行（模拟“uniq”）
# 只保留重复行中的第一行，其他行删除

sed '$!N; /^\(.*\)\n\1$/!P; D'
```

```
awk '{if($0!=B)print;B=$0}'
```

删除文件中的重复行，不管有无相邻。注意 hold space 所能支持的缓存大小，或者使用 GNU sed。

```
sed -n 'G; s/\n/&&/; /^\([ -~]*\n\).*\n\1/d; s/\n//; h; P'    #bones7456 注：我这里此命令并不能正常工作
awk '{if(!($0 in B))print;B[$0]=1}'
```

删除除重复行外的所有行（模拟“uniq -d”）

```
sed '$!N; s/^\(.*\)\n\1$/\1/; t; D'
awk '{if($0==B && $0!=l){print;l=$0}B=$0}'
```

删除文件中开头的 10 行

```
sed '1,10d'
awk '{if(NR>10)print}'
```

删除文件中的最后一行

```
sed '$d'
```

#awk 在过程中并不知道文件一共有几行，所以只能通篇缓存，大文件可能不适合，下面两个也一样

```
awk '{B[NR]=$0}END{for(i=0;i<=NR-1;i++)print B[i]}'
```

删除文件中的最后两行

```
sed 'N;$!P;$!D;$d'
awk '{B[NR]=$0}END{for(i=0;i<=NR-2;i++)print B[i]}'
```

删除文件中的最后 10 行

```
sed -e :a -e '$d;N;2,10ba' -e 'P;D'     # 方法 1
sed -n -e :a -e '1,10!{P;N;D;};N;ba'    # 方法 2
awk '{B[NR]=$0}END{for(i=0;i<=NR-10;i++)print B[i]}'
```

删除 8 的倍数行

```
gsed '0~8d'                                    # 只对 GNU sed 有效
sed 'n;n;n;n;n;n;n;d;'                         # 其他 sed
awk '{if(NR%8!=0)print}' |head
```

删除匹配式样的行

```
sed '/pattern/d'                    # 删除含 pattern 的行。当然 pattern 可以换成任何有效的正则表达式
awk '{if(!match($0,/pattern/))print}'

# 删除文件中的所有空行（与“grep ‘.’ ”效果相同）

sed '/^$/d'                         # 方法 1
sed '/./!d'                         # 方法 2
awk '{if(!match($0,/^$/))print}'

# 只保留多个相邻空行的第一行。并且删除文件顶部和尾部的空行。
# （模拟“cat -s”）

sed '/./,/^$/!d'          #方法 1，删除文件顶部的空行，允许尾部保留一空行
sed '/^$/N;/\n$/D'        #方法 2，允许顶部保留一空行，尾部不留空行
awk '{if(!match($0,/^$/)){print;F=1}else{if(F)print;F=0}}'   #同上面的方法 2

# 只保留多个相邻空行的前两行。

sed '/^$/N;/\n$/N;//D'
awk '{if(!match($0,/^$/)){print;F=0}else{if(F<2)print;F++}}'

# 删除文件顶部的所有空行

sed '/./,$!d'
awk '{if(F || !match($0,/^$/)){print;F=1}}'

# 删除文件尾部的所有空行

sed -e :a -e '/^\n*$/{$d;N;ba' -e '}'   # 对所有 sed 有效
sed -e :a -e '/^\n*$/N;/\n$/ba'         # 同上，但只对 gsed 3.02.*有效
awk '/^.+$/{for(i=l;i<NR-1;i++)print "";print;l=NR}'

# 删除每个段落的最后一行

sed -n '/^$/{p;h;};/./{x;/./p;}'

#很长，很 ugly，应该有更好的办法

awk -vRS= '{B=$0;l=0;f=1;while(match(B,/\n/)>0){print
substr(B,l,RSTART-l-f);l=RSTART;sub(/\n/,"",B);f=0};print ""}'
```

特殊应用：

```
# 移除手册页（man page）中的 nroff 标记。在 Unix System V 或 bash shell 下使
# 用’echo’命令时可能需要加上 -e 选项。
```

```
sed "s/.`echo \\\b`//g"      # 外层的双括号是必须的（Unix 环境）
sed 's/.^H//g'               # 在 bash 或 tcsh 中，按 Ctrl-V 再按 Ctrl-H
sed 's/.\x08//g'             # sed 1.5，GNU sed，ssed 所使用的十六进制的表示方法
awk '{gsub(/.\x08/,"",$0);print}'

# 提取新闻组或 e-mail 的邮件头

sed '/^$/q'                  # 删除第一行空行后的所有内容
awk '{print}/^$/{exit}'

# 提取新闻组或 e-mail 的正文部分

sed '1,/^$/d'                # 删除第一行空行之前的所有内容
awk '{if(F)print}/^$/{F=1}'

# 从邮件头提取“Subject”（标题栏字段），并移除开头的“Subject:”字样

sed '/^Subject: */!d; s///;q'
awk '/^Subject:.*/{print substr($0,10)}/^$/{exit}'

# 从邮件头获得回复地址

sed '/^Reply-To:/q; /^From:/h; /./d;g;q'

#好像是输出第一个 Reply-To:开头的行？From 是干啥用的？不清楚规则。。

awk '/^Reply-To:.*/{print;exit}/^$/{exit}'

# 获取邮件地址。在上一个脚本所产生的那一行邮件头的基础上进一步的将非电邮地址的部分剃除。（见上一脚本）

sed 's/ *(.*)//; s/>.*//; s/.*[:<] *//'

#取尖括号里的东西吧？

awk -F'[<>]+' '{print $2}'

# 在每一行开头加上一个尖括号和空格（引用信息）

sed 's/^/> /'
awk '{print "> " $0}'

# 将每一行开头处的尖括号和空格删除（解除引用）

sed 's/^> //'
awk '/^> /{print substr($0,3)}'
```

```
# 移除大部分的 HTML 标签（包括跨行标签）

sed -e :a -e 's/<[^>]*>//g;/</N;//ba'
awk '{gsub(/<[^>]*>/,"",$0);print}'

# 将分成多卷的 uuencode 文件解码。移除文件头信息，只保留 uuencode 编码部分。
# 文件必须以特定顺序传给 sed。下面第一种版本的脚本可以直接在命令行下输入；
# 第二种版本则可以放入一个带执行权限的 shell 脚本中。（由 Rahul Dhesi 的一
# 个脚本修改而来。）

sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode     # vers. 1
sed '/^end/,/^begin/d' "$@" | uudecode                      # vers. 2

#我不想装个 uudecode 验证，大致写个吧

awk '/^end/{F=0}{if(F)print}/^begin/{F=1}' file1 file2 ... fileX

# 将文件中的段落以字母顺序排序。段落间以（一行或多行）空行分隔。GNU sed 使用
# 字元“\v”来表示垂直制表符，这里用它来作为换行符的占位符——当然你也可以
# 用其他未在文件中使用的字符来代替它。

sed '/./{H;d;};x;s/\n/={NL}=/g' file | sort | sed '1s/={NL}=//;s/={NL}=/\n/g'
gsed '/./{H;d};x;y/\n/\v/' file | sort | sed '1s/\v//;y/\v/\n/'
awk -vRS= '{gsub(/\n/,"\v",$0);print}' ll.txt | sort | awk '{gsub(/\v/,"\n",$0);print;print ""}'

# 分别压缩每个.TXT 文件，压缩后删除原来的文件并将压缩后的.ZIP 文件
# 命名为与原来相同的名字（只是扩展名不同）。（DOS 环境：“dir /b”
# 显示不带路径的文件名）。

echo @echo off >zipup.bat
dir /b *.txt | sed "s/^\(.*\)\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat

DOS 环境再次略过，而且我觉得这里用 bash 的参数 ${i%.TXT}.zip 替换更帅。

下面的一些 SED 说明略过，需要的朋友自行查看原文。

{ Source. Thanks bones7456. }

原帖地址：http://linuxtoy.org/archives/sed-awk.html
```