Unix-like Operating Systems

Sed and awk.

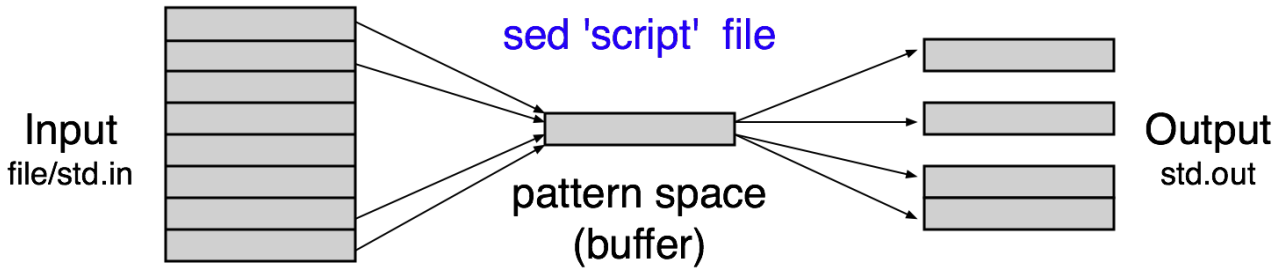
Jan Trdlička

Czech Technical University in Prague, Faculty of Information Technology
Department of Computer Systems

Content

Filter sed

```
sed [options] 'script' [files]
sed [options] -e 'script' [files]
sed [options] -f script_file [files]
```



- Stream editor - reads lines from standard input/input files, makes editing changes according to a script of editing commands, and writes the results to standard output.
- Useful options
  - `-e` *script* ... add the script to the commands to be executed.
  - `-f` *script_file* ... add the contents of file to the commands to be executed.
  - `-n` ... suppress automatic printing of pattern space.
  - `-E` ... use ERE rather than BRE (only GNU).

Filter sed: script and commands

- A script consists of commands, one per line, of the following form:

  `[ address1] [ , address2 ] ] command [ arguments ]`

- Address
  - *m* ... input line with sequential number *m*.
  - `$` ... the last line of input.
  - */RE/* ... a context address (line which consists of a /*RE*/).
  - *m*,*n* ... inclusive range of input lines from *m* to *n*.
  - */RE1/*,*/RE2/* ... inclusive range, from a line containing /*RE*1/ to a line containing /*RE*2/.
- Useful commands
  - `-s`*/RE/REP/*`flags` ... substitute the string *REP* for an instances of the /RE/ in the pattern space (input lines).
    - Flag *n* ... substitute for just the *n*-th instance ($1 \leq n \leq 512$).
    - Flag `g` ... substitute for all instances.
  - `-d` ... delete pattern space (input lines).
  - `-p` ... print the current pattern space (input lines).

Filter sed: examples of printing specific lines (by address)

- Print all input lines to the standard output (default behaviour).

```
ls -l / | nl | sed ' '
```

```
ls -l / | nl | sed -n 'p'
```

- Print the first 5 lines by command p or d.

```
ls -l / | nl | sed -n '1,5p'
```

```
ls -l / | nl | sed '6,$d'
```

- Print lines 5, ...10 by command p or d.

```
ls -l / | nl | sed -n '5,10p'
```

```
ls -l / | nl | sed -e '1,4d' -e '11,$d'
```

- Print the last five lines by command p or d.

```
ls -l / | nl > /tmp/f          # remember the output of ls
```

```
sed -n  "$(( $( wc -l < /tmp/f ) - 5 )),\$p" /tmp/f
```

```
sed "1,$(( $( wc -l < /tmp/f ) - 6 ))d" /tmp/f
```

```
rm /tmp/f                      # remove temporary file
```

Filter sed: examples of text substitution

- How to replace the first occurrence of the string "ABC" with the string "___"?

```
echo "ABC hjj kkj ABC hjhk abc uhkh ABC kjABCkj" | \
  sed -E 's/ABC/___/'
```

- How to replace the second occurrence of the string "ABC" with the string "___"?

```
echo "ABC hjj kkj ABC hjhk abc uhkh ABC kjABCkj" | \
  sed -E 's/ABC/___/2'
```

- How to replace all occurrences of the string "ABC" with "___"?

```
echo "ABC hjj kkj ABC hjhk abc uhkh ABC kjABCkj" | \
  sed -E 's/ABC/___/g'
```

- How to replace all occurrences of the strings "ABC" or "abc" with "___"?

```
echo "ABC hjj kkj ABC hjhk abc uhkh ABC kjABCkj" | \
  sed -E 's/ABC|abc/___/g'
```

Filter sed: examples of text substitution

- How to print all suffixes of words from file `/usr/share/lib/dict/words` on `fray1.fit.cvut.cz`, that start with the string `"work"`.
  - Find words beginning with the string `work` and having a suffix.

```
grep -E '^work.+' words
```

  - Get suffixes.

```
grep -E '^work.+' words | sed -E 's/^work(.*)/\1/'
```

Filter sed: examples of text substitution

- How to replace the first word on a line with string "FIRST" (words consist only of alphabetic characters)?
  - Verifying a regular expression by command grep.

```
man bash | grep -E '^[^[:alpha:]]*\<[[:alpha:]]*\>'
```

  - Replacement of a regular expression by command sed.

```
man bash | \
  sed -E 's/^([^[:alpha:]]*)\<[[:alpha:]]*\>/\1FIRST/'
```

Filter sed: examples of script

- How to make the following modification by `sed`
  - Delete the lines starting with string "total".
  - Append string "<- - link" to the end of the line beginning with the letter 'l'.
  - Replace "Oct" with "OCT" and "Nov" with "NOV".
- Create a script `m.sed` that contains the commands for `sed`.

```
/^total/d
s/^l.*$/&    <-- link/
s/Oct/OCT/
s/Nov/NOV/
```

- Run the script.

```
ls -l /etc | sed -f m.sed
```

Filter sed: examples of printing specific lines (by pattern)

- Assume that we have a xml file `f.xml` with the following contents.

```
<person>
   <name>
      Peter
   </name>
   <age>
      25
   </age>
</person>
<person>
   <name>
      Susan
   </name>
   <age>
      31
   </age>
</person>
```

- How to print only names from the file `f.xml`?

  Solution 1

```
sed -n '/<name>/,/<\/name>/p' f.xml | grep -Ev '<name>|</name>'
```

Filter sed: examples of printing specific lines (by pattern)

- How to print only names from the file `f.xml`?

  Solution 2

  - Get one record of person per line.

```
<person><name>Peter</name><age>25</age></person>
<person><name>Susan</name><age>31</age></person>
```

```
tr -d ' \n' < f.xml | sed -E 's:</person>:</person>\n:g'
```

    - Note: In substitution `'s/RE1/RE2/g'`, you can use any character as an RE separator to avoid confusion with the character in RE (e.g. `'s:RE1:RE2:g'`).

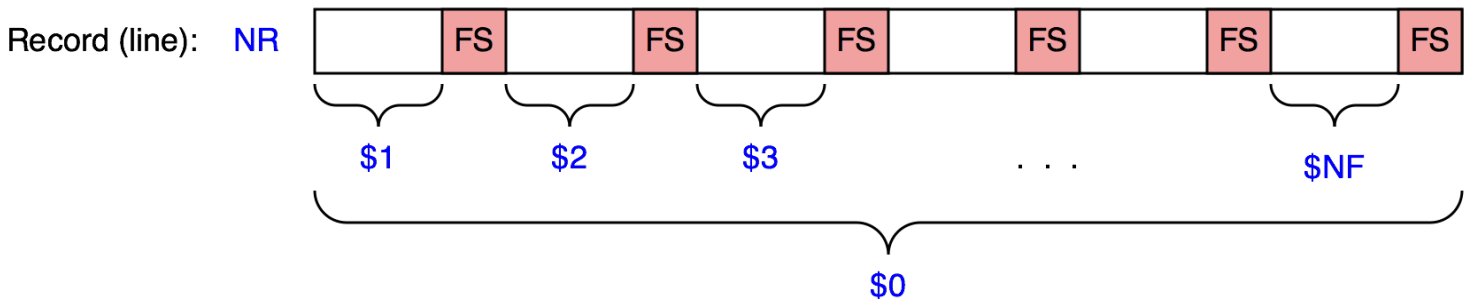
  - Delete all characters before the name and after the name.

```
tr -d ' \n' < f.xml | sed -E 's:</person>:</person>\n:g'|\
  sed -E 's:^.*<name>(.*)</name>.*$:\1:'
```

Filter awk

`awk [options]` *program_text* `[file]`
`awk [options] -f` *program_file* `[file]`

- Awk = Aho, Weinberger, Kernighan.
- Useful options
  - `-f` *program_file* ... read the program from the file.
  - `-F` *fs* ... use *fs* as the input field separator (space, TAB by default).
  - `-v` *var=val* ... assign the value *val* to the variable *var*.
- The filter `awk` reads records (lines) from the standard input/input file, makes editing changes according to a program.
- Each record is split into fields, using the value of field separator FS.
- Values of fields are available by built-in awk variables: $0, $1,..., $NF.



Filter awk: examples of splitting input lines

- For each item of directory `/etc`, print its name and size in bytes.

```
ls -la /etc | tail -n+2 | awk '{print $9 ":\t" $7 " B"}'
```

```
ls -la /etc | tail -n+2 | awk '{printf("%s:\t%d B\n", $9, $7)}'
```

- Print names of users who have an account on `fray1.fit.cvut.cz`.
  Hint: Use the 5th field of output of command `getent passwd`.

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd ' | \
  awk -F':' '{ print $5 }'
```

- Print only family names from the previous examaple.
  - Why the following solution doesn't work?

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd ' | \
  awk -F':' '{ print $5 }' | awk '{ print $(NF-1) }'
```

  - How to get correct solution?

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd ' | \
  awk -F':' '{ print $5 }' | awk '{ print NF , $0 }' | \
  grep '[^ ]* [^ ]*' | awk '{print $(NF-1)}'
```

Filter awk: program structure

`[ pattern ] [ { code } ]`

- Pattern `BEGIN`: The code is executed before the first input record is read.
- Pattern `END`: The code is executed after the all the input is exhausted.
- `pattern`: Code is executed for every input record. If a `pattern` is defined, then the code is performed only for records (lines) that conform to the pattern.
  - Regular expression `/RE/`: ERE is supported.
  - Logical expression:
    - Similar to C language.
    - Relational operators: $<$, $>$, $<=$, $>=$, $==$, $!=$, $\sim$, $!\sim$.
    - Mathematical operators: $+$, $-$, $*$, $/$, $\%$, $\hat{}$, $++$, $--$.
    - Logical operators: $\&\&$, $||$, $!$, ( ).
- `code`:
  - Similar to C language.
  - Condition statement: `if ( ) { } else { }`.
  - Loops: `for ( ) { }` , `while () { }`.

Filter awk: examples of printing specific lines

- Print list of symbolic links in directory /etc.

```
~> cat code1.awk
BEGIN {
   print "List of soft links";
   print "---------------";
}
/^l/ {
   print "\t" $0;
}
END {
   print "---------------";
}
```

```
ls / | awk -f code1.awk
```

- Print lines number 3, ..., 5 from /etc/passwd.

```
awk ' NR >= 3 && NR <= 5 { print $0} ' /etc/passwd
```

```
awk '{ if ( NR >= 3 && NR <= 5 )  print $0}' /etc/passwd
```

- Print line from /etc/passwd that represents an account with UID=0 (the 3rd item on line).

```
awk -F':' '$3 == 0 { print $0 }' /etc/passwd
```

```
awk -F':' '{ if ( $3 == 0)  print $0 }' /etc/passwd
```

Filter awk: variables

- Variables are dynamic (they come into existence when they are first used).
- It is strongly recommended to initialise every user variable.
- Variables types
  - Simple types: string, floating-point number.
  - Associative arrays.
- Built-in variables
  - `RS` ...the input record separator(newline by default).
  - `FS` ... the input field separator (space, TAB by default).
  - `NF` ... the number of fields in the current input record (line).
  - `NR` ... the total number of input records (lines) seen so far.
  - `$0` ... the value of the record (line).
  - `$1`, ... , `$NF` ... the values of fields.

Filter awk: examples of variables

- How many running processes are started by user `root` and how much memory are they using?
  Hint: Use command `ps -eo ruser,rss,cmd`.

```
~> cat code2.awk
BEGIN {
  count = 0;
  size  = 0;
}
/^ *root / {
  count++;
  size += $2;
}
END {
  printf("Cout = %d processes  Size = %d kB\n", count, size);
}
```

```
ps -eo ruser,rss,cmd | awk -f code2.awk
```

- Which user account on `fray1.fit.cvut.cz` represents a user with the largest number of names?

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd' | \
   awk -F':' '{print $5}' | \
   awk ' NF>2 {print NF , $0}' |
   sort -k1,1n | tail -2
```

Filter awk: built-in functions

- Numeric functions
  `sin( )`, `cos( )`, `log( )`, `rand( )`, ...
- String functions
  `printf()`, `length()`, `match()`, `split()`, `gsub()`, `substr()`, ...
- Other useful functions
  `system()`

Filter awk: examples of built-in functions

- Which user account on `fray1.fit.cvut.cz` represents a user with the longest family name?

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd' | \
  awk -F':' '{print $5}' | \
  awk ' NF>2 {print length($(NF-1)) , $0}' | \
  sort -k1,1n | tail -1
```

```
~> cat code3.awk
BEGIN {
   FS        = ":";
   account   = "";
   maxLength = 0;
}
{
   n = split($5, a, " ");
   l = length(a[n-1])
   if ( l > maxLength ) {
     account   = $0;
     maxLength = l;
   }
}
END {
  printf("%s\n", account)
}
```

```
ssh $USER@fray1.fit.cvut.cz 'getent passwd' | awk -f code3.awk
```