Programming in shell 1

Filters
and
useful Unix commands.

Jan Trdlička

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

Content

Filters

- Filter is a "simple" program that gets its data from its standard input (the main input stream) and writes its results to its standard output (the main output stream).
- Examples: `head`, `tail`, `wc`, `cut`, `tr`, ...
- Filters are often used as elements of pipelines.
  - Which process allocates the most memory?

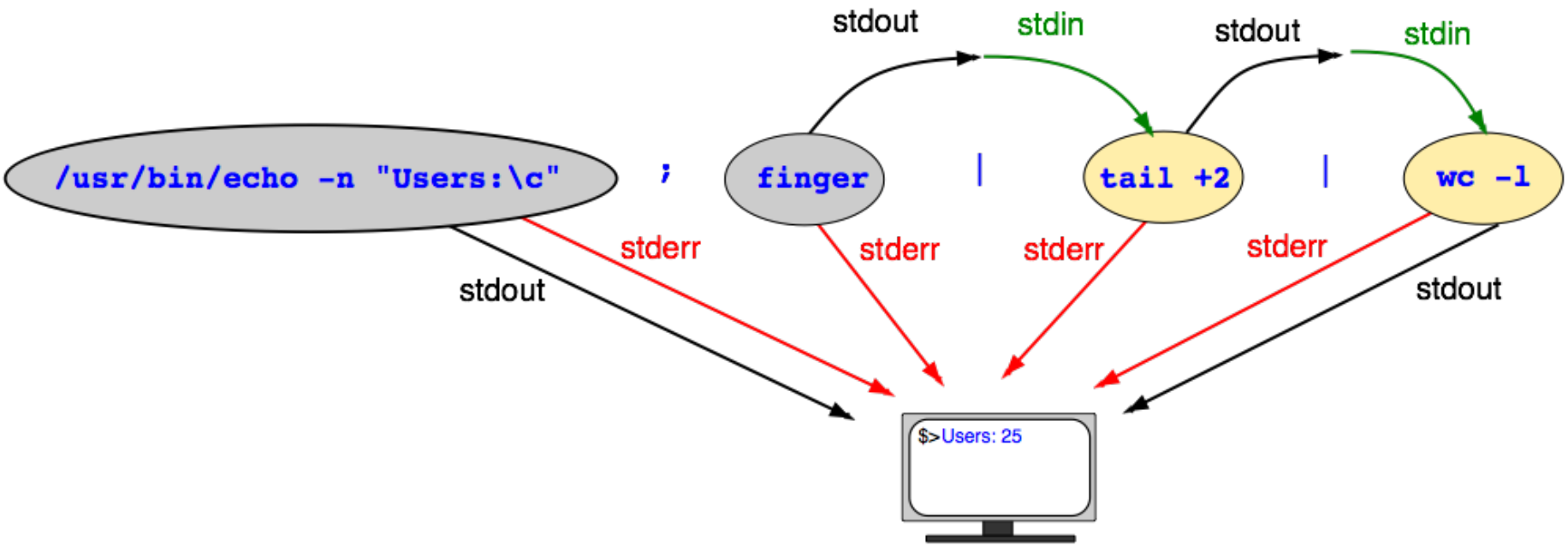
```
ps -e -o rss,user,pid,comm | sort -k1,1n | tail -1
```

  - Which students are currently logged in to the server fray1.fit.cvut.cz?

```
finger -lf | grep 'student$' | tr -s '\t ' | \
  cut -d' ' -f8- | sed 's/student$//'
```

- Why use Unix filters and not my own C program?
  - It is not proprietary solution.
  - Anyone can simply modify the solution.
  - Platform portability.

Every application is not filter



tee

`tee [options] [files]`

- The filter reads lines from the standard input and writes them to the standard output and files.
- Options
  - `-a` ... appends the output to the files.
- Examples
  - How many items (files, directories, ...) are in the directory `/etc`?
    - Save list of items to the file `items.txt`.
    - Print the number of items to the standard output.

```
ls /etc | tee items.txt | wc -l
```

nl

`nl [options] [files]`

- The filter numbers lines of the standard input/files and prints them to the standard output.
- Options
  - `-s'`*sep*`'` ... *sep* is the character(s) used in separating the line number.
  - `-bp'`*pattern*`'` ... only lines containing the *pattern* will be numbered.
- Examples
  - Default line numbering.

```
ls -l /etc | nl
```

  - New separator between number and original line.

```
ls -l /etc | nl -s') '
```

  - Only lines with patter will be numbered.

```
ls -l /etc | nl -bp'^-'
```

wc

`wc [options] [files]`

- The filter prints a count of lines, words and characters of standard input/files to the standard output.
- Options
  - `-c` ... counts bytes.
  - `-w` ... counts words.
  - `-l` ... counts lines.
- Examples
  - How many files (items) are in the working directory?

```
ls -a | wc -w
ls -a | wc -l
```

  - How many user accounts are on the server `fray2.fit.cvut.cz`?

```
ssh trdlicka@fray2.fit.cvut.cz 'getent passwd | wc -l'
```

tr

`tr [options] set1 set2`

- The filter copies the standard input to the standard output with substitution or deletion of selected characters.
- The *set*1 and *set*2 operands control translations that occur while copying characters.
- Options
  - `-c` ... use the complement of *set*1.
  - `-d` ... delete all occurrences of characters that are specified by *set*1.
  - `-s` ... replace instances of repeated characters with a single character.
- Supported meta-characters
  - Ranges (depends on the locale)
    - *M*-*N* ... all of the characters from *M* through *N* (GNU).
    - [*M*-*N*] ... all of the characters from *M* through *N* (System V).
  - Character classes
    - [:*CLASS*:] ... expands to all of the characters in the class *CLASS*.
  - Repeated characters
    - [*C***N*] ... in *set*2 expands to *N* copies of character *C*.
    - [*C**] ... expands to as many copies of *C* as are needed to make *set*2 as long as *set*1.

tr

- Examples
  - Replace the following characters: a → X, b → Y, and c → Z, in the output of the command `ls -l /`.

```
ls -l / | tr 'abc' 'XYZ'
```

  - Replace lower case by upper case in the output of `ls -l /`.

```
ls -l / | tr 'a-z'          'A-Z'          # GNU Linux
ls -l / | tr '[a-z]'        '[A-Z]'        # Solaris

ls -l / | tr '[:lower:]'  '[:upper:]' # locale independent
```

  - Replace all characters with a underscore character except characters *a* through *z* and newline in the output of the command `ls -l /`.

```
ls -l / | tr  -c '[:lower:]\n' '[_*]'
```

  - Modify the output of the command `ls -l /` so that adjacent columns are separated by just one space.

```
ls -l  | tr -s ' '
```

head

`head [options] [files]`

- The filter prints first 10 lines of standard input/files to the standard output.
- Options
  - -*k* ... copies the first *k* lines from standard input to standard output.
  - -n-*k* ... copies all lines except the last *k* lines from standard input to standard output (GNU implementation).
- Examples
  - List names of the five largest files from the working directory.

```
ls -S | head -5
```

  - List names of the three files from the working directory whose content has been last modified.

```
ls -t | head -3
```

tail

`tail [options] [file]`

- The filter prints the last lines of the standard input/file to the standard output.
- Options
  - `-k` ... begins printing at k-th item from end of file.
  - `+k` ... begins printing at k-th item from beginning of file, `-n+k` ... GNU implementation.
  - `-f` ... dosn't quit at the end of file (use CTRL-C to quit).
- Examples
  - List names of the five smallest files from the working directory.

```
ls -S | tail -5
```

  - List the five largest files from the working directory including details.

```
ls -lS | tail -n+2 | head -5     # GNU Linux
ls -lS | tail +2 | head  -5      # Solaris
```

  - Execute the following commands in two different terminals.

```
date > /tmp/f ; tail -f /tmp/f
```

```
for (( i=0;i<5;i++)); do sleep 2; date >> /tmp/f ; done
```

cut

`cut [options] [files]`

- The filter cuts out selected fields of each line of the standard input/file and prints them to the standard output.
- Options
  - `-c` *list* ... specifies characters (e.g. 2-10,15,45-).
  - `-d` *delim* ... defines the field delimiter (-f option only).
  - `-f` *list* ... specifies fields separated in the file by a delimiter character.
- Examples
  - For each file in your working directory, list its access rights and name

```
ls -l | cut -c2-10,54-    # Attribute dependent solution
```

```
ls -l | tr -s ' ' | cut -d' ' -f1,9 | cut -c2-
```

  - For each user account on the server `fray2.fit.cvut.cz`, print the account name (the first item) and user information (the fifth item).

```
ssh trdlicka@fray2.fit.cvut.cz \
   'getent passwd | cut -d":" -f1,5'
```

paste

`paste [options] files`

- The utility merges corresponding or subsequent lines of files and print them to the standard output.
- Options
  - `-d`*list* ... each character in list is an element specifying a delimiter character.
  - `-s` ... concatenate all of the lines of each separate input file in command line order.
- Examples
  - Save the name, uid and shell of users who have an account on this server to the files `/tmp/name`, `/tmp/uid` and `/tmp/shell`, respectively.

```
getent passwd | cut -d":" -f1 > /tmp/name.txt
getent passwd | cut -d":" -f3 > /tmp/uid.txt
getent passwd | cut -d":" -f7 > /tmp/shell.txt
```

  - Merge the previous files so that each row contains: `uid+shell*name`.

```
paste -d"+*" /tmp/uid.txt /tmp/shell.txt /tmp/name.txt
```

paste

- Examples
  - Write every third line from the list of the command `ls /etc | nl`

```
ls /etc | nl | paste -s -d'::\n' | cut -d':' -f3
```

join

`join [options] file1 file2`

- For each pair of input lines with identical join fields (common lines), write a line to standard output.
- Utility requires sorted input files.
- Options
  - `-t` *delim* ... defines the field delimiter.
  - `--header` ... treat the first line in each file as field headers.
  - `-1` *M* ... join on the filed *M* of file `file1`.
  - `-2` *N* ... join on the field *N* of file `file2`.
  - `-a` *N* ... print unpairable lines from `fileN`.
  - `-o auto` ... useful when dealing with unpaired lines.
  - `-e` *TEXT* ... replace missing input fields with *TEXT*.
  - `-v` *N* ... like `-a` , but suppress common lines.

join

- Examples

`file1`

Name	Age
Anna	18
Bob	27
Peter	23
Sophia	71
Susan	4
Tom	53

`file2`

Name	Height	Weight
Anna	162	56
Bob	180	87
Henry	175	98
Peter	169	72
Susan	179	70
Tom	183	101

`file3`

Height	Weight	Name
162	56	Anna
180	87	Bob
175	98	Henry
169	72	Peter
179	70	Susan
183	101	Tom

- Join files `file1` and `file2` by name (common lines).

```
join --header file1 file2
```

- Join files `file1` and `file3` by name (common lines).

```
join --header -1 1 -2 3 file1 file3
```

- Join files `file1` and `file3` by name (all lines).

```
join --header -1 1 -2 3 -a1 -a2 -o auto -e NA file1 file3
```

split

`split [options] file [prefix]`

- The utility splits a file into pieces of given size with given names: *prefix*aa, *prefix*ab, *prefix*ac, ...
- Options
  - `-b` *n* ... splits a file into pieces *n* bytes in size.
  - `-l` *n* ... splits a file into pieces *n* lines in size.
  - `-a` *n* ... *n* is length of name suffix.
- Examples
  - Split file `/bin/date` into 10kB pieces (files).

```
split -b10k /usr/bin/date date
```

  - Merge the previous pieces to file `mydate` .

```
cat date?? > mydate
```

  - Split file `/etc/passwd` into 5 line pieces (files).

```
split -l 5 -a 3 /etc/passwd passwd
```

  - Merge the previous pieces to file `mypasswd`.

```
cat passwd??? > mypasswd
```

sort

`sort [options] [files]`

- The filter sorts lines of all the named files together and writes the result on the standard output.
- Options
  - `-f` ... folds lower-case letters into upper case.
  - `-n` ... sorts in arithmetic order.
  - `-M` ... compares as months.
  - `-r` ... reverses the sense of comparisons.
  - `-u` ... identical lines in input file appear only one (uniq).
  - `-t`*char* ... uses *char* as the field separator character.
  - `-k`*start_field*`[.`*start_char*`][,`*end_field*`[.`*end_char*`]]` ... restricted sort key field definition.

sort

- Examples
  - Sort the output of `ls -l /` alphabetically.

```
ls -l / | sort
```

  - Sort the output of `ls -l /` alphabetically by the sixth column.

```
ls -l / | sort -k6,6
```

  - Sort the output of `ls -l /` by the sixth column as month.

```
ls -l / | sort -k6,6M
```

  - Sort the output of `ls -l /` by the fifth column as number.

```
ls -l / | sort -k5,5n
```

  - Sort the output of `ls -l /` by the date and after by size.

```
ls -l / | sort -k6,6M -k7,7n -k5,5n
```

  - Sort the output of `ls -l /` by the time.

```
ls -l / | sort  -k8.2,8.3n -k8.5,8.6n
```

uniq

`uniq [options] [file]`

- The filter reports or filters out repeated lines in a file and print them to the standard output.
- Options
  - `-c` ... precedes each output line with a count of the number of times.
- Examples
  - Which users have an application running on this server?

```
ps -eo user | tail -n+2 | sort | uniq
```

```
ps -eo user | tail -n+2 | sort -u       # better solution
```

  - How many processes have each user running?
    (Frequency table: users x number of processes)

```
ps -eo user | tail -n+2 | sort | uniq -c
```

cmp

`cmp [options] file1 file2`

- The utility compare two files byte by byte.
- Options
  - `-s` ... writes nothing for differing files and returns only exit status.
- Examples
  - Create two files by the following commands.

```
printf "%s\n" a b c d e f > f1.txt
printf "%s\n" a c "new line" d "e modified line" f > f2.txt
```

  - Compare the previous files.

```
cmp f1.txt f1.txt
cmp f1.txt f2.txt
```

  - Compare the previous files and print only string "Same", if they are identical.

```
cmp -s f1.txt f1.txt && echo "Same"
cmp -s f1.txt f2.txt && echo "Same"
```

comm

`comm [options] file1 file2`

- The utility reads `file1` and `file2`, which must be ordered in the current collating sequence, and produces three text columns as output.
  1. lines only in `file1`,
  2. lines only in `file2`,
  3. and lines in both files.
- Options
  - `-1` ... suppress column 1 (lines unique to `file1`).
  - `-2` ... suppress column 2 (lines unique to `file2`).
  - `-3` ... suppress column 3 (lines that appear in both files).
- Examples
  - Compare previous files.

```
comm f1.txt f2.txt
```

  - List rows that are the same in both files.

```
sort f1.txt > f1.sort
sort f2.txt > f2.sort
comm -12 f1.sort f2.sort
```

diff

`diff [options] file1 file2`

- The utility compares two files.
- Options
  - `-u` ... produces a listing of differences with lines of context.
    - `+` ... lines added or changed in `file2`.
    - `-` ... removed and changed lines in `file1`.

- Examples
  - Compare previous files.

```
diff -u f1.txt f2.txt
```

xargs

`xargs [options] [command]`

- The utility build and execute command lines from standard input.
- Options
  - `-I` *replstr* ... utility taking the entire line as a single argument, inserting it in argument for each occurrence of *replstr*.
- Examples
  - Create the following files and directory.

```
touch {a,b,c}.{png,c,tar,gz,txt,jpg} ; mkdir pictures
```

  - Move all files with suffix `.png` or `.jpg` to the directory `pictures`

```
printf "%s\n" *.png *.jpg | xargs -I FILE mv FILE pictures
```