

# Unix-like Operating Systems

Filters  
and  
useful Unix commands.

Jan Trdlička



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

- 1 Filters – definition
- 2 Useful filters
  - tee, nl, wc, tr
- 3 Filters/utilities for text splitting and merging
  - head, tail
  - cut, paste
  - split, cat
- 4 Filters for sorting and duplicity search
  - sort
  - uniq
- 5 File comparison tools
  - cmp, comm, diff
- 6 Filter for building and executing commands from standard input
  - xargs

- **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?

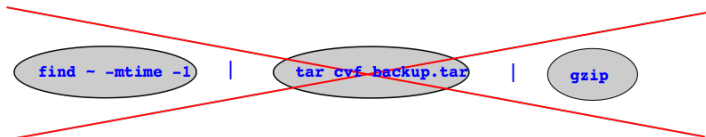
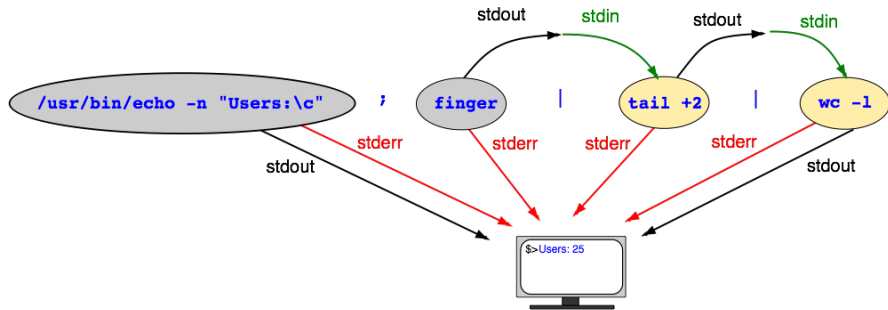
```
ps -e -o rss,user,pid,comm | sort -k1,1n | tail -1
```

- 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 [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 [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 [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 [options] set1 set2

- The filter copies the standard input to the standard output with substitution or deletion of selected characters.
- The *set1* and *set2* operands control translations that occur while copying characters.
- Options
  - **-c** ... use the complement of *set1*.
  - **-d** ... delete all occurrences of characters that are specified by *set1*.
  - **-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 *set2* expands to *N* copies of character *C*.
    - ***[C\*]*** ... expands to as many copies of *C* as are needed to make *set2* as long as *set1*.



## Examples

- Replace the following characters:  $a \rightarrow X$ ,  $b \rightarrow Y$ , and  $c \rightarrow Z$ , 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** ... doesn'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
```

- Examples

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

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

# split

`split [options] file [prefix]`

- The utility splits a file into pieces of given size with given names:  
*prefixaa, prefixab, prefixac, ...*

- **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 [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).
  - `-tchar` ... uses *char* as the field separator character.
  - `-kstart_field[.start_char][,end_field[.end_char]]` ... restricted sort key field definition.



## • 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 [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 [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 [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 [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 [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
```