



---

# Lecture 4

Unix: Filters and useful commands.

*Department of Computer Systems FIT, Czech Technical University in Prague*  
*©Jan Trdlička, 2011*



- Filtr
  - It is a “simple” program that gets most of its data from its standard input (the main input stream) and writes its main results to its standard output (the main output stream).
  - Examples. head, tail, wc, cut, tr, ...
- UNIX filters are often used as elements of pipelines.
  - `ps -e -o rss,user,comm | sort -k1,1n | tail -1`
  - `du -ks /home/stud/* | sort -k1,1nr | head -10 | sed 's/\/.*\///'`
- Why use UNIX filters and not my own C program?
  - It is not proprietary solution
  - Anyone can simply modify the solution
  - Platform portability



```
tee [-ai] [files]
```

- The tee utility will copy standard input to standard output, making a copy in zero or more files.
  - a Appends the output to the files
  - i Ignores interrupts

- **Examples:**

```
finger | tee a.txt b.txt
```

```
spell text.txt | tee -a typos
```



`nl [options] [files]`

- Line numbering filter.
  - s'sep' sep is the character(s) used in separating the line number
  - bp'pattern' Specifies which lines are to be numbered.

- **Examples:**

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

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

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



```
head [-n] [file]
```

- Display first n lines of files (n=10 is default).

- **Examples:**

```
ls -lct / | nl | head -4
```



```
tail [options] [file]
```

- Print the last lines of file.

- n      begin printing at nth item from end of file
  - +n      begin printing at nth item from beginning of file
  - f      don't quit at the end of file
  - r      copy lines in reverse order

- **Examples:**

```
ls -lct / | nl | tail
```

```
ls -lct / | nl | tail -5
```

```
ls -lct / | nl | tail +5
```

```
tail -f file.txt
```



`wc [options] [files]`

- Display a count of lines, words and characters in files
  - c counts bytes
  - w counts words
  - l counts lines

- **Examples:**

```
ls -l / | tail +2 | wc -l
```

```
wc -l /etc/passwd
```



## cut options [files]

- Cut out selected fields of each line of a file.
  - c *list* specifies character s (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:**

```
ls -l | cut -c2-10,15-23,54-
```

```
ypcat passwd | cut -d: -f1,3,5
```

```
who | cut -d' ' -f1
```





`paste [options] files`

- Merge corresponding or subsequent lines of files
  - `-dlist` each character in list is an element specifying a delimiter character

- **Examples:**

```
ypcat passwd | cut -d: -f1 > name.txt
```

```
ypcat passwd | cut -d: -f3 > uid.txt
```

```
ypcat passwd | cut -d: -f7 > shell.txt
```

```
paste -d'+-' uid.txt name.txt shell.txt
```



`split [options] file name`

Split a file into pieces of given size with given name: `nameaa`, `nameab`, `nameac`,...

- b size      splits a file into pieces n bytes in size
- l size      splits a file into pieces n lines in size
- a length    length of name suffix

- Examples:**

```
split -b 10k /bin/ls ls
```

```
cat ls?? > ls
```

```
man ls | nl > ls.txt
```

```
split -l 10 -a 5 ls.txt ls
```

```
cat ls????? > ls.txt
```



`uniq [options] [file]`

- Report or filter out repeated lines in a file.
  - c Precedes each output line with a count of the number of times the line occurred in the input.
  - d Suppresses the writing of lines that are not repeated in the input.
  - u Suppresses the writing of lines that are repeated in the input.

- **Examples:**

```
who | cut -d" " -f1 | sort > names.txt
```

```
uniq -c names.txt
```

```
uniq -d names.txt
```

```
uniq -u names.txt
```



`tr [options] string1 string2`

- Copies the standard input to the standard output with substitution or deletion of selected characters. The `string1` and `string2` operands
  - d Deletes all occurrences of input characters that are specified by `string1`.
  - s Replaces instances of repeated characters with a single character. `znakem`

- **Examples:**

```
cat file.txt | tr 'ABC' 'xyz'
```

```
cat file.txt | tr '[a-z]' '[A-Z]'
```

```
ls -l | tr -s ' ' | cut -d' ' -f2,9-
```

```
ls -l | tr -s ' ' '\012'
```



`sort [options] [files]`

- Sorts lines of all the named files together and writes the result on the standard output
  - f folds lower-case letters into upper case
  - n sort 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* use *char* as the field separator character.
  - k-*start\_field* [*type*][,*end\_field*[*typ*]e]  
restricted sort key field definition



## Examples:

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

```
ls -l / | sort -k3
```

```
ls -l / | sort -k3,3
```

```
ls -l / | sort -k2,2n
```

```
ls -l / | sort -k2,2nr
```

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

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

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

```
ls -l / | sort -k6,6M -k7,7n -k8.2,8.3n -k8.5,8.6n
```

```
ypcat passwd | sort -t':' -k3,3n
```



```
cmp [options] file1 file2
```

- Compare two files
  - s write nothing for differing files;  
return exit status only

- **Examples:**

```
$ cmp s1.txt s3.txt
```

```
s1.txt s3.txt differ: char 15, line 2
```



```
comm [-123] file1 file2
```

- The comm utility reads file1 and file2, which must be ordered in the current collating sequence, and produces three text columns as output: lines only in file1; lines only in file2; and lines in both files.

## Examples:

```
comm s1.txt s3.txt
```

```
comm -12 s1.txt s3.txt
```





```
diff [options] file1 file2
```

- Compare two files .
- The normal output contains lines of these forms:

n1 **a** n3,n4

n1,n2 **d** n3

n1,n2 **c** n3,n4

where n1 and n2 represent lines file1 and n3 and n4 represent lines in file2 (a,d,c are command for editor ed).

-u produces a listing of differences with three lines of context(+, -).

+ lines added or changed in file2

- removed and changed lines in file1

-c produces a listing of differences with three lines of context. (+,-,!).



```
patch [options] [files]
```

- The patch command reads a source (patch) file containing any of the three forms of difference (diff) listings produced by the diff(1) command (normal, context or in the style of ed(1)) and apply those differences to a file.

-b	Saves a copy of the original contents of each modified file(file.orig)
-i patchfile	Reads the patch information from the file patchfile
-R	Reverses the sense of the patch script

## Examples:

```
diff -u f1 f2 > patch.diff
```

```
patch -b -i patch.diff f1
```

```
patch -b -R -i patch.diff f1
```



## `find directory [expression]`

- The find utility recursively descends the directory hierarchy for each path seeking files that match a Boolean expression.
- Parts of expression can be
  - grouped by `\ ( a \ )`
  - negated with `\ !`
  - separated by logical and `-a` (default) , logical or `-o` .



## . Actions

- print  
printed. Always true. Causes the current pathname to be printed.
- exec cmd { } \; True if the executed command returns a zero value as exit status.
- ok cmd { } \; Like -exec, but it requires confirmation.
- ls Always true. Prints current pathname together with its associated statistics.

## . Predicates

- type [d,f,l,b,c] true if the type of the file is d,f,l,b,c
- inum n true if the file has inode number n
- name 'vzor' true if pattern matches the basename (use shell spec. character \*,?,[ ] ,...)
- perm nnn true if file has given access permissions
- mtime [n | -n | +n] true if the file's data was modified n days ago
- atime [n | -n | +n] true if the file was accessed n days ago
- ctime [n | -n | +n] true if the file's status was changed n days ago



## Examples:

```
find $HOME
```

```
find $HOME -type f
```

```
find /bin -type f -name at
```

```
find /usr/bin -type f -name '*grep'
```

```
find . -atime -1
```

```
find . -name '[A-Z]*'
```

```
find . \! -name '[A-Z]*'
```



## Examples:

```
find . -name 'f*' -print
```

```
find . -name 's*' -print
```

```
find . \( -name 'f*' -o -name 's*' \) -print
```

```
find . \! \( -name 'f*' -o -name 's*' \) -print
```

```
find . -print -name 'f*' -print
```

```
find . -name core -ok rm { } \;
```

```
find . -name core -exec rm { } \;
```



- Construct argument lists and invoke utility.
- **Examples:**

```
mkdir ~/sources
```

```
find ~ -type f -name "*.c" | xargs -I {} -t mv {} ~/sources/
```

