# Linux recap and Basic Tools

Oleg O. Brovko

*ICTP, CMSP | LB.266 | obrovko@ictp.it*

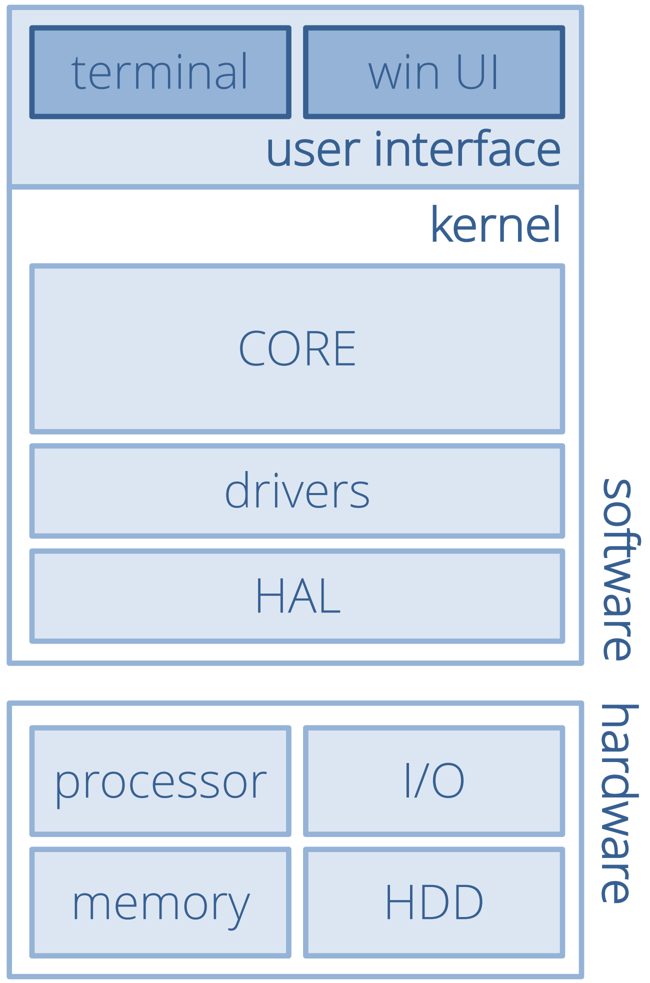
In brief:

- *intro to living in Linux*
- *command prompt / console / IO streams*
- *basic navigation / file management*
- *simple data processing*
- *plotting*

Trieste, October 2017

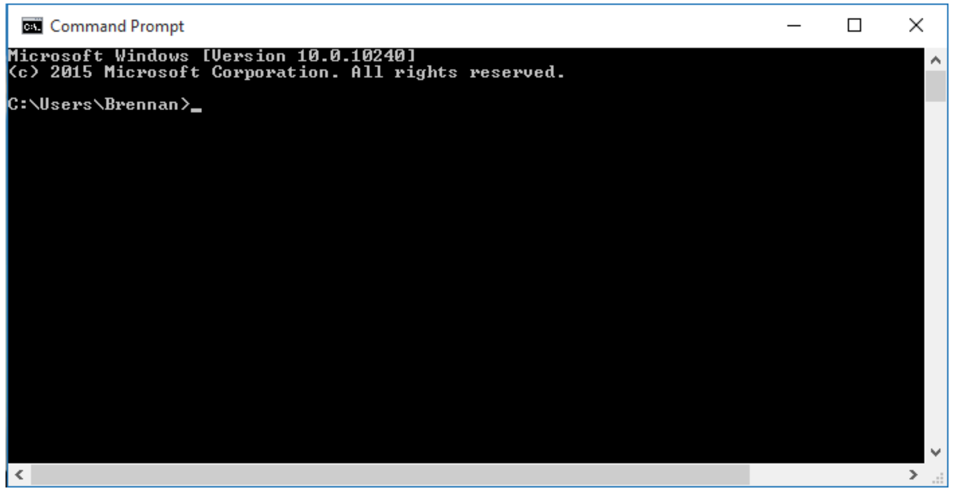
## Oversimplified computer architecture

- PC
  - hardware
    - processor, memory, HDD
    - IO devices: keyboard, screen, audio…
  - software
    - kernel (computation, hardware interface, etc)
    - user interface (terminal, windows)

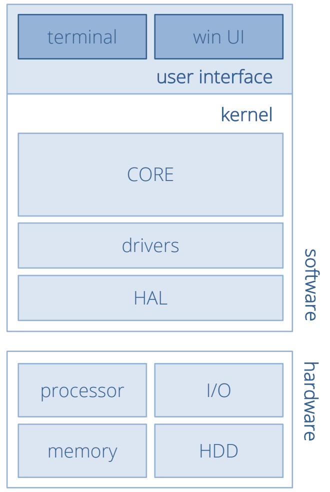


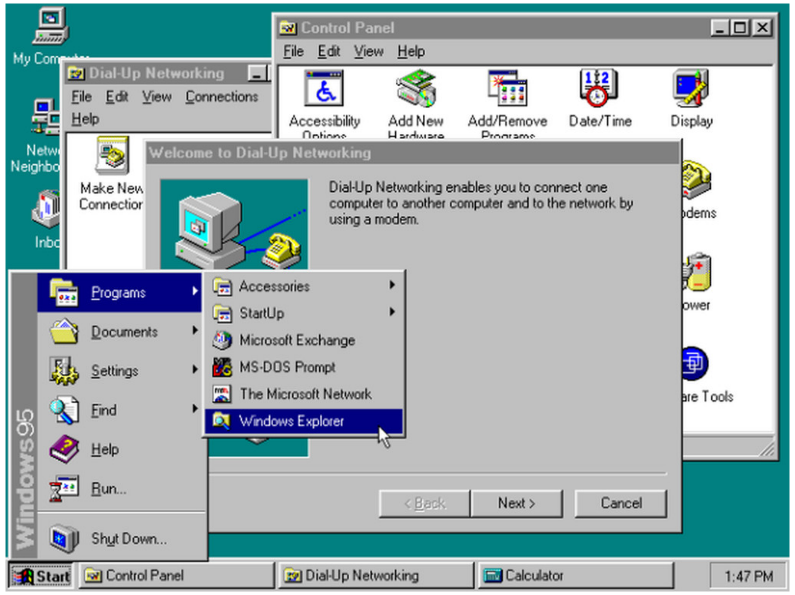
## Windows also has a command prompt…

- MS Windows
  - core: win 9x, win NT
  - user interface
    - windows interface (9x, XP, NT, Vista, 8, 10)
    - command prompt



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Brennan>_
```

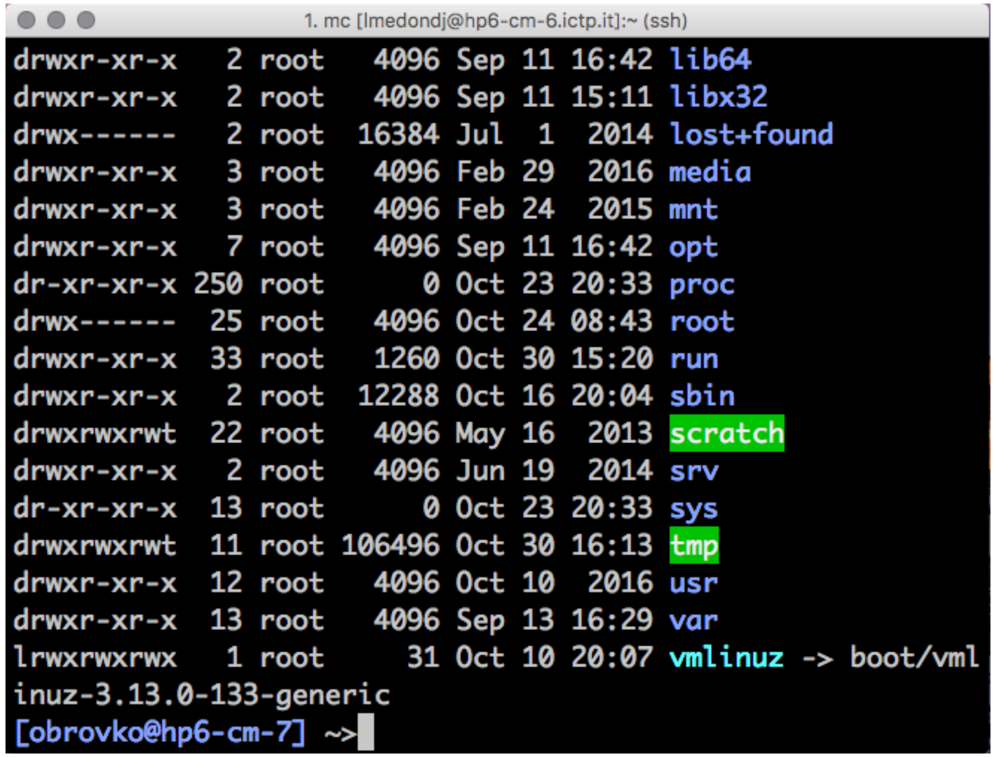




Dial-Up Networking enables you to connect one computer to another computer and to the network by using a modem.

## Linux: multitude of windows and console platforms...

- Linux
  - core: kernel (currently v 3.xx)
  - user interface
    - console (sh, bash, csh, tcsh)
    - x11, xorg + KDE / Gnome / xfce ...



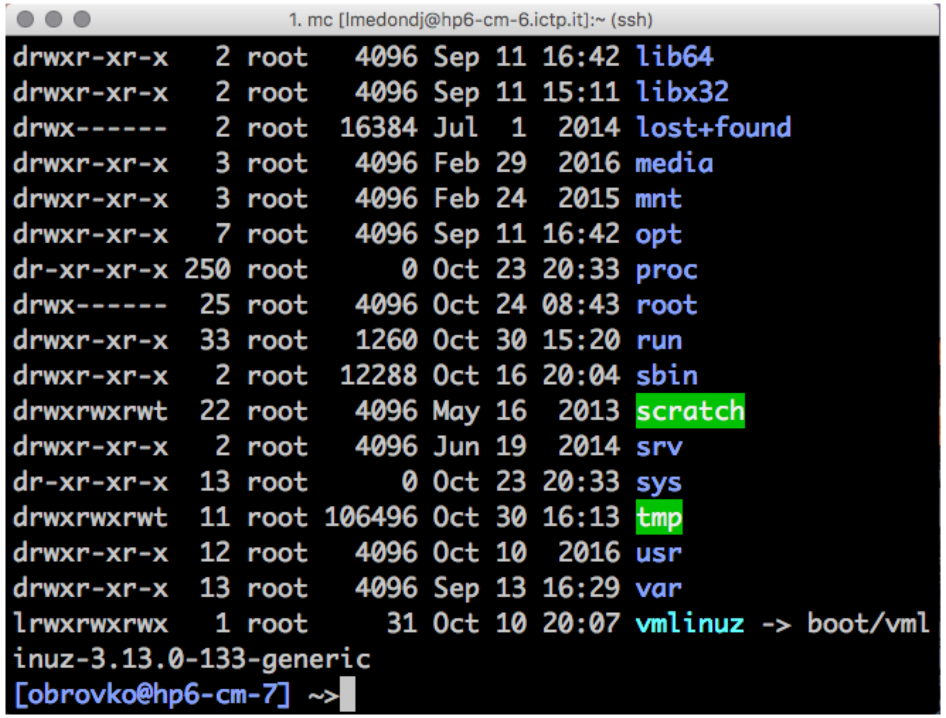
```
drwxr-xr-x   2 root    4096 Sep 11 16:42 lib64
drwxr-xr-x   2 root    4096 Sep 11 15:11 libx32
drwx------   2 root   16384 Jul  1  2014 lost+found
drwxr-xr-x   3 root    4096 Feb 29  2016 media
drwxr-xr-x   3 root    4096 Feb 24  2015 mnt
drwxr-xr-x   7 root    4096 Sep 11 16:42 opt
dr-xr-xr-x 250 root       0 Oct 23 20:33 proc
drwx------  25 root    4096 Oct 24 08:43 root
drwxr-xr-x  33 root    1260 Oct 30 15:20 run
drwxr-xr-x   2 root   12288 Oct 16 20:04 sbin
drwxrwxrwt  22 root    4096 May 16  2013 scratch
drwxr-xr-x   2 root    4096 Jun 19  2014 srv
dr-xr-xr-x  13 root       0 Oct 23 20:33 sys
drwxrwxrwt  11 root  106496 Oct 30 16:13 tmp
drwxr-xr-x  12 root    4096 Oct 10  2016 usr
drwxr-xr-x  13 root    4096 Sep 13 16:29 var
lrwxrwxrwx   1 root      31 Oct 10 20:07 vmlinuz -> boot/vml
inuz-3.13.0-133-generic
[obrovko@hp6-cm-7] ~>
```

## User interfaces...

- Console
  - text input/output
  - several terminals available
    - bash, csh, tcsh, sh (process terminal commands)
    - do not confuse with terminal GUI (xterm, konsole, etc...)
- Prompt
  - displays metainfo, current folder (customizable)
    - `[username@hostname] path>`
    - `hostname:path>`
- Input

```
> command list_of_parameters special_directives
```

  - space separated
  - commands
    - built-in terminal commands
    - program name (system or user written – no difference)
  - parameters
    - **`options`** to programm
    - **`file/directory names`**


1. mc [lmedondj@hp6-cm-6.ictp.it]:~ (ssh)

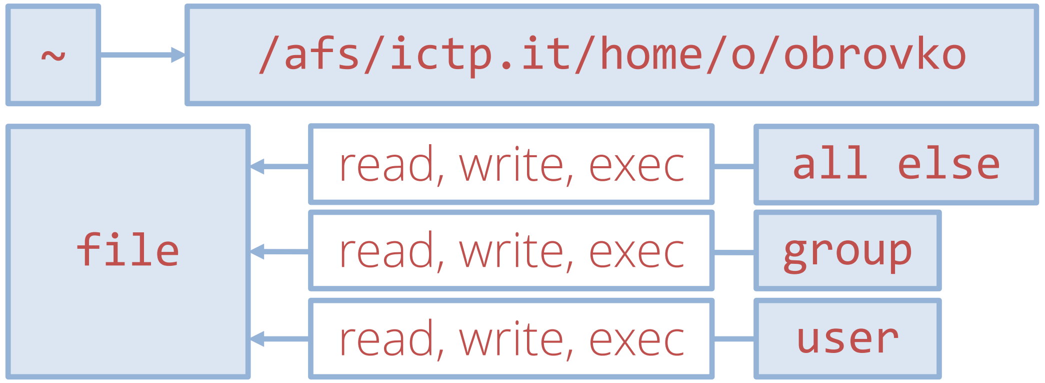
```
drwxr-xr-x    2 root    4096 Sep 11 16:42 lib64
drwxr-xr-x    2 root    4096 Sep 11 15:11 libx32
drwx------    2 root   16384 Jul  1  2014 lost+found
drwxr-xr-x    3 root    4096 Feb 29  2016 media
drwxr-xr-x    3 root    4096 Feb 24  2015 mnt
drwxr-xr-x    7 root    4096 Sep 11 16:42 opt
dr-xr-xr-x  250 root       0 Oct 23 20:33 proc
drwx------   25 root    4096 Oct 24 08:43 root
drwxr-xr-x   33 root    1260 Oct 30 15:20 run
drwxr-xr-x    2 root   12288 Oct 16 20:04 sbin
drwxrwxrwt   22 root    4096 May 16  2013 scratch
drwxr-xr-x    2 root    4096 Jun 19  2014 srv
dr-xr-xr-x   13 root       0 Oct 23 20:33 sys
drwxrwxrwt   11 root  106496 Oct 30 16:13 tmp
drwxr-xr-x   12 root    4096 Oct 10  2016 usr
drwxr-xr-x   13 root    4096 Sep 13 16:29 var
lrwxrwxrwx    1 root      31 Oct 10 20:07 vmlinuz -> boot/vml
inuz-3.13.0-133-generic
[obrovko@hp6-cm-7] ~>
```

```
> cd ~/numI/ass04
> cp prog03.f90 prog04.f90
> gfortran -o prog04 prog04.f90
```

# Linux filesystem structure

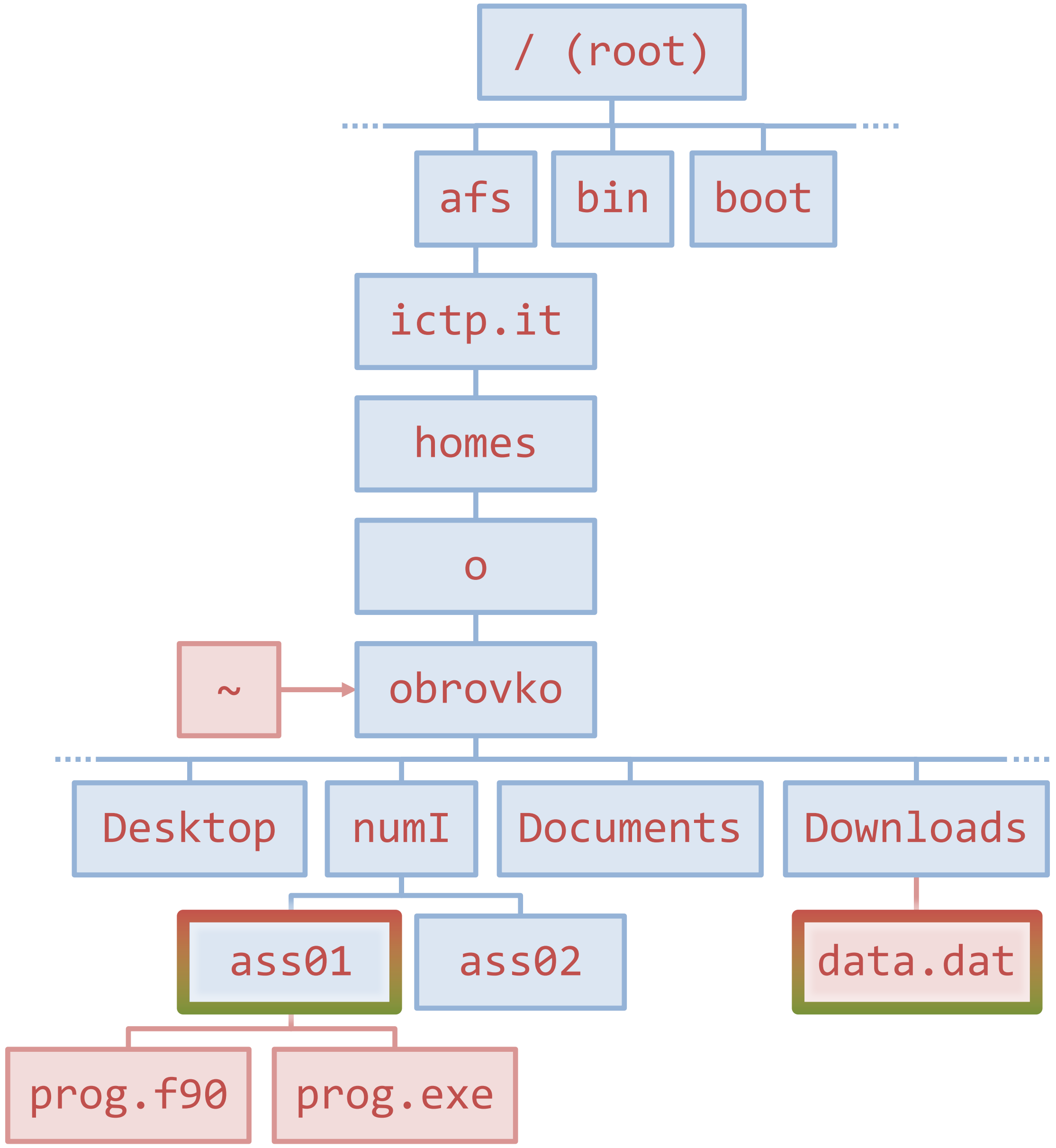
- Windows
  - drives (`C:\`), folders, files, shortcuts
  - path delimeter: `'\'` : `C:\Users\obrovko\cat.jpg`
- Linux / Unix
  - root `'/'`
  - subfolders, files, links
  - mount points
    - network filesystems `/afs/ictp/home/o/obrovko`
    - disks `/media/obrovko/usb_stick`
  - path delimeter: `'/'`
    - `/afs/ictp.it/home/o/obrovko/ass04.f90`
  - `'~'` points to home folder)
    - `~/ass04.f90`
  - on UNIX FS: file access control
    - separate for user, group, all (read, write, execute)
  - filename characters
    - in theory – any symbols, except `/, NUL`
    - steer clear of special characters: `? * + % ~`

```
/ (root)
├ home      - local user homes
├ afs       - network FS
├ media     - mount points for media
├ boot      - boot files
├ dev       - device handles
├ lib       - libraries
├ proc      - runtime info
├ run       - runtime files
├ scratch   - temporary local disk
├ usr       - non-kernel software
├ etc       - system setting files
└ var       - log files, etc…
```



## Finding files...

- Paths
  - absolute (starting from `'/'`)
  - relative (to `current path` or home `'~'`)
- Special characters
  - `~` – user home folder
  - `.` – current folder
  - `..` – parent folder
- Examples
  - current `/afs/ictp.it/homes/o/obrovko/numI/ass01`
  - `.` ⟼ `ass01`
  - `..` ⟼ `numI`
  - `../..` ⟼ `obrovko`
  - `~` ⟼ `obrovko`
  - address `data.dat`
    - `/afs/ictp.it/homes/o/obrovko/Downloads/data.dat`
    - `../../Downloads/data.dat`
    - `~/Downloads/data.dat`
    - `~/numI/ass01/../../Download/data.dat`



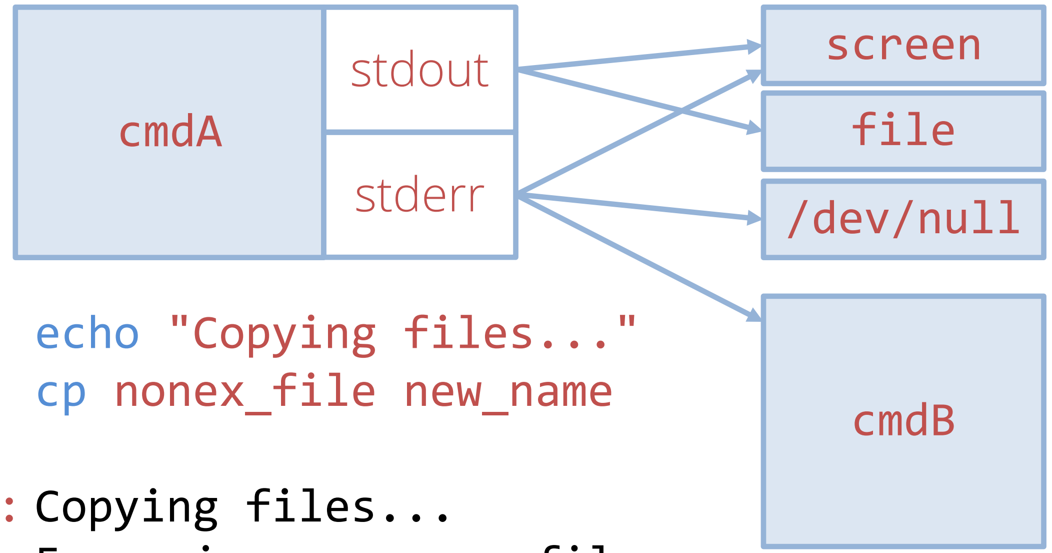
# bash – the basics

- bash
  - a terminal
  - scripting/programming language
- As programming language
  - variables
  - logical constructs
  - loops
  - functions
- Script
  - text file
  - sequence of commands
  - can executed as a function (with parameters)

```
#!/bin/bash

myState="counting"
yourNumber="12"

echo "I am $myState till $yourNumber"

for i in 1 2 3
do
    echo "I am $myState $i"
done

if [[ -e data.log ]]
then
    cp data.log newdata.log
    echo "Renamed data.log..."
fi
```

# bash – Understanding output

- understanding output
  - output streams
    - standard output
    - standard error
  - devices
    - screen
    - file
    - garbage collector / void
  - redirect `stdout` (normal output)
    - `cmdA params >outfile.txt`
    - `cmdA params 1>outfile.txt`
  - redirect `stderr` (errors go here)
    - `cmdA params 2>errfile.txt`
  - overwrite *vs* append
    - `cmdA params >out.txt` shall overwrite out
    - `cmdA params >>out.txt` shall append to the end
  - redirect into void
    - `cmdA params 1>output.dat 2>/dev/null`



```
echo "Copying files..."
cp nonex_file new_name
```

```
stdout: Copying files...
stderr: Error in cp: nonex_file:
        No such file or directory
```

- redirect to another process
  - pipe operator
    - `cmdA params | cmdB`

## bash – useful perks

- autocomplete
  - Tab key – bash tries to guess the command or filename
  - 2xTab – print available options
- history
  - keys up and down browse through recent commands
  - Ctrl+r allows to search the whole of ~/.bash_history
- RTFM
  - help on commands and programs
    - man xxx
    - scroll, search with /,n and exit with q
    - info yyy (alternative)
  - google for examples if in doubt
- wildcards in filenames
  - * shall match any character sequence in filenames and paths
  - given files out_1.dat, out_2.dat, out.log, prog
  - rm out* shall remove all out files and rm out*.dat only the first 2
- all variables and wildcards are expanded before execution

```
out_1.dat
out_2.dat
prog

> logfile="rm.log"

> rm *.dat 2>$logfile
> rm log_1.dat log_2.dat 2>out.log
```

## Script files in bash

- Scripts
  - programs not compiled to binary
  - executed by interpreter line by line
- bash
  - `# starts a comment`
  - script starts with interpreter choice (optional) `#!/bin/bash` (could be, f.e., `python` or `perl`)
  - lines are executed sequentially
  - loops and logical tests (see `basics.sh`)
  - use indentation to structure code!
  - `\` at the end of a line to continue command
- Execution
  - `./script_name`
  - `. script name` (mind the space)
  - `. relative/path/to/file`
  - `. /absolute/path/to/file`
  - `bash filename_or_path`
- Notes
  - for the script to be run it needs have read and execute permissions set for the user

```
- rwx r-x ---
  usr grp all
```

  - change permissions with `chmod` `chmod u+x`
    - `u/g/a` – permission for user/group/all
    - `+/-` – set/remove permission
    - `r/w/x` – read/write/execute
  - `./scriptname` runs in a separate bash instance
    - active environment not affected
    - variables are not saved
    - current directory not changed
  - `. scriptname` runs in the same instance
    - same as just execute the lines one by one
    - changes local environment

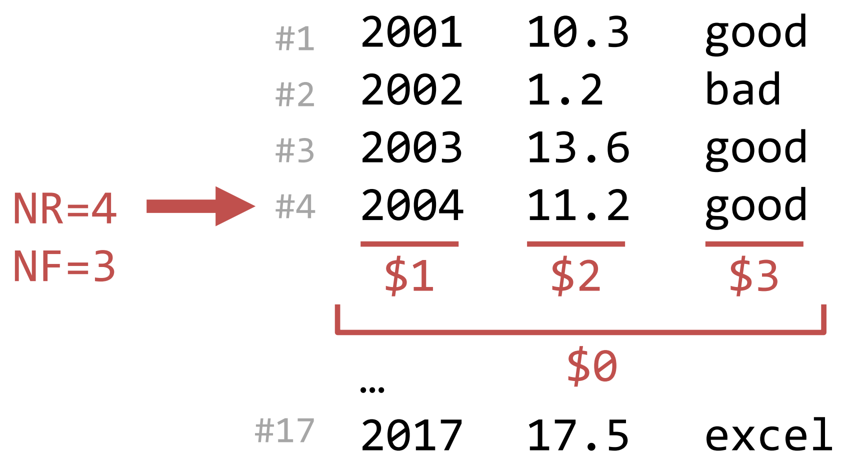
# Basic sed

- Stream Editor
  - find and replace in files or stream
  - `sed 's/pattern/replace/' myfile`
  - `sed -E 's/regexp/replace/modifiers' myfile > outfile`
- RegEx
  - regular expression
  - powerful search and replace language
  - google for syntax
- Modifiers
  - `s/find/rep/` – replace first occurrence
  - `s/find/rep/g` – all occurrences
  - `s/find/rep/I` – case insensitive
- As a stream editor
  - `echo 'Hello, my dear world!' | sed 's/Hello/Bye/' | sed 's/my dear/damned/'`
  - `echo 'Myyyy keeeeys are stickkkkyyy!' | sed -E 's/([a-z])\1+/\1/' > outfile.dat`

## Basic awk

- awk (Aho, Weinberger, and Kernighan)
- Logic
  - go through file or stream line by line
  - split each line into "fields" and process according to rules
- Variables
  - `NR` – line currently being processed
  - `NF` – number of fields
  - `$0` – the whole line
  - `$1, $2, …` – field values
- Operations
  - arithmetic, string operations
  - `printf / print` (formatted / unformatted)
- Conditions
  - numerical `(NR>10 && $1<15){…}`
  - string match `($3 ~ /ad/){…}`
  - apply to all lines `{}`

```
awk 'BEGIN{action_start};
    (condition1){action1};
    (condition2){action2}
    END{action_end}' myfile
```



```
#1  2001  10.3  good
#2  2002  1.2   bad
#3  2003  13.6  good
#4  2004  11.2  good
…
#17 2017  17.5  excel
```