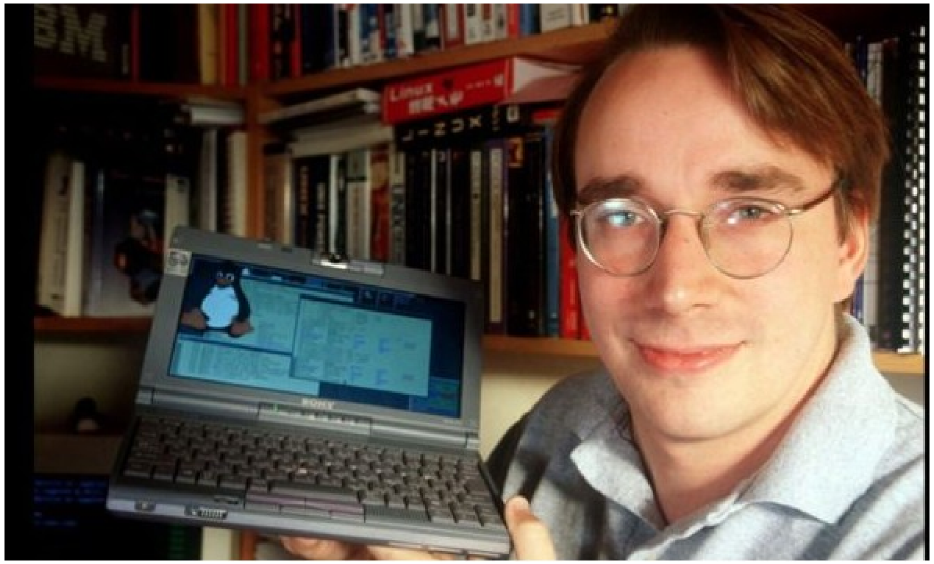
Linux intro. The Shell

Linux and Unix

Photo: Alcatel-Lucent

Key Figures: Ken Thompson [seated] types as Dennis Ritchie looks on in 1972, shortly after they and their Bell Labs colleagues invented Unix.

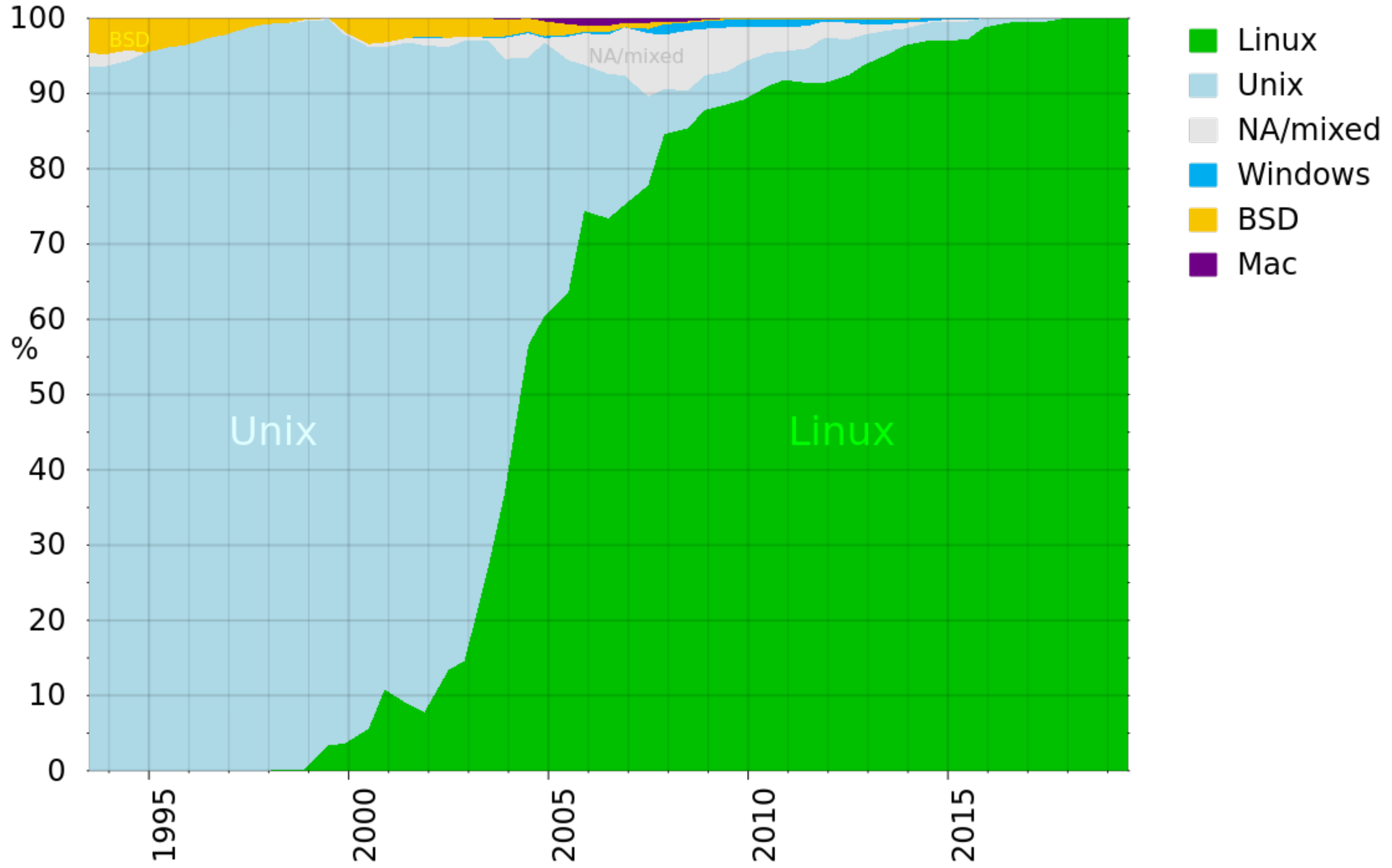


Linus Torwalds creator of Linux kernel in early 1990s

Where Linux is used

aws Azure Google Cloud

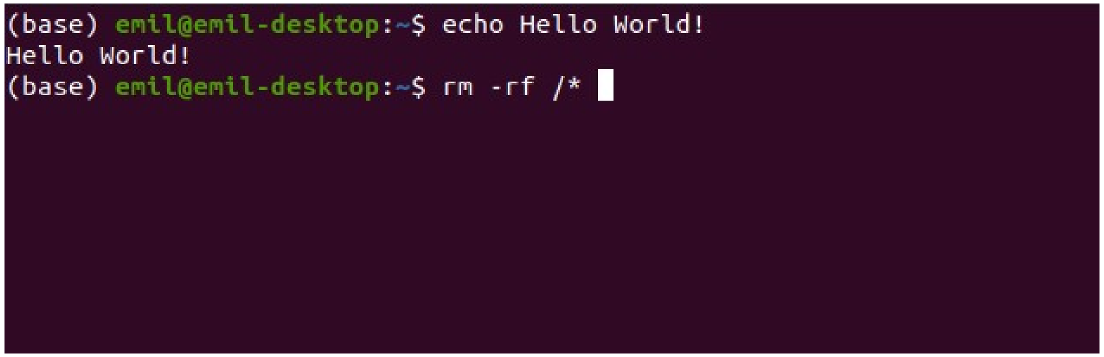
Top 500 supercomputers use Linux



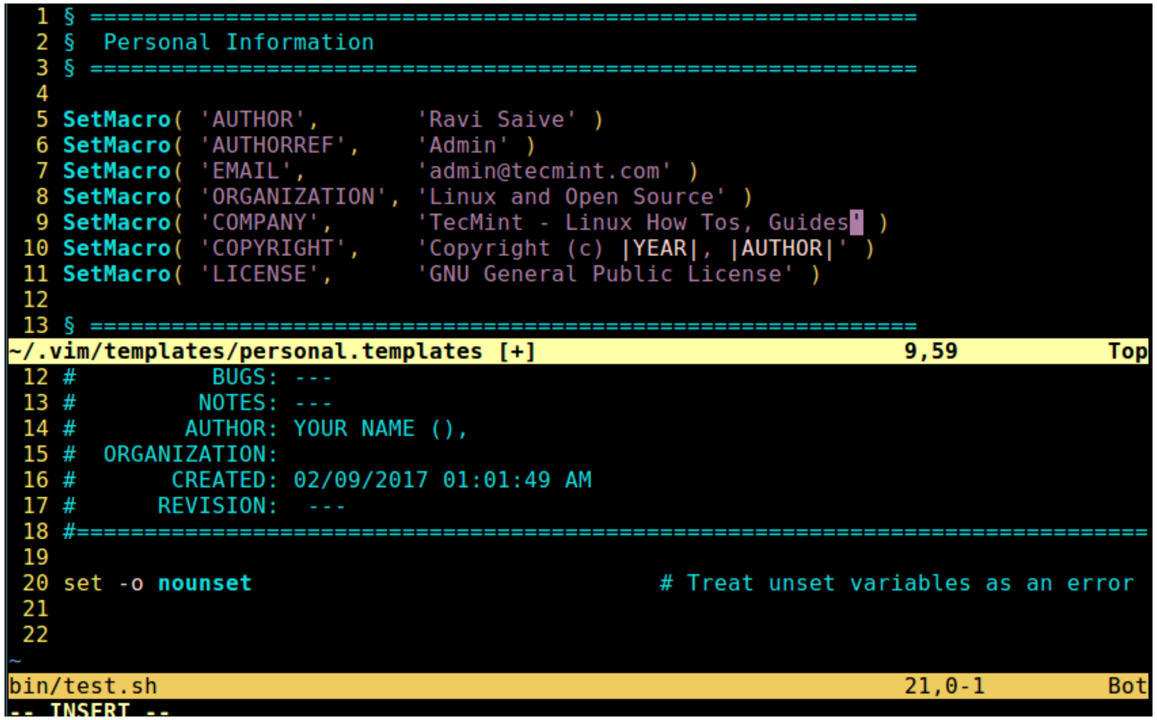
The power of Linux

- heavy computations on clusters vs analysis and development on local desktops
- flexibility in software management
- possibility to participate in open source software

The Shell



```
(base) emil@emil-desktop:~$ echo Hello World!
Hello World!
(base) emil@emil-desktop:~$ rm -rf /*
```



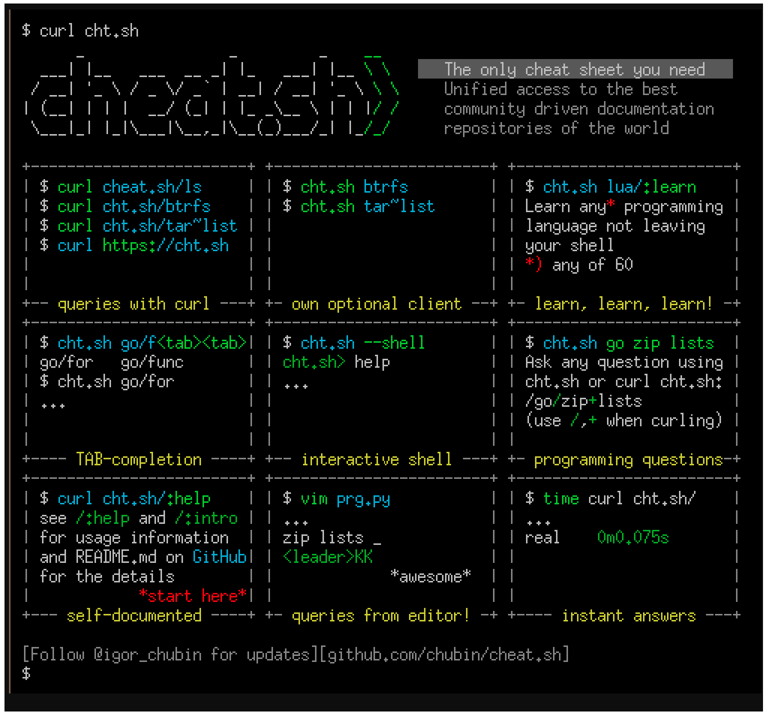
```
§ ==============================================================
§  Personal Information
§ ==============================================================

SetMacro( 'AUTHOR',       'Ravi Saive' )
SetMacro( 'AUTHORREF',    'Admin' )
SetMacro( 'EMAIL',        'admin@tecmint.com' )
SetMacro( 'ORGANIZATION', 'Linux and Open Source' )
SetMacro( 'COMPANY',      'TecMint - Linux How Tos, Guides' )
SetMacro( 'COPYRIGHT',    'Copyright (c) |YEAR|, |AUTHOR|' )
SetMacro( 'LICENSE',      'GNU General Public License' )

§ ==============================================================
~/.vim/templates/personal.templates [+]                    9,59          Top
#          BUGS: ---
#         NOTES: ---
#        AUTHOR: YOUR NAME (),
#  ORGANIZATION:
#       CREATED: 02/09/2017 01:01:49 AM
#      REVISION:  ---
#==========================================================================

set -o nounset                              # Treat unset variables as an error


~
bin/test.sh                                                21,0-1        Bot
-- INSERT --
```



```
$ curl cht.sh
      _                _         _    __
  ___| |__   ___  __ _| |_   ___| |__ \ \      The only cheat sheet you need
 / __| '_ \ / _ \/ _` | __| / __| '_ \ \ \     Unified access to the best
| (__| | | |  __/ (_| | |_ _\__ \ | | |/ /     community driven documentation
 \___|_| |_|\___|\__,_|\__(_)___/_| |_/_/      repositories of the world

+------------------------+ +------------------------+ +------------------------+
| $ curl cheat.sh/ls     | | $ cht.sh btrfs         | | $ cht.sh lua/:learn    |
| $ curl cht.sh/btrfs    | | $ cht.sh tar~list      | | Learn any* programming |
| $ curl cht.sh/tar~list | |                        | | language not leaving   |
| $ curl https://cht.sh  | |                        | | your shell             |
|                        | |                        | | *) any of 60           |
|                        | |                        | |                        |
+-- queries with curl ---+ +- own optional client --+ +- learn, learn, learn! -+
+------------------------+ +------------------------+ +------------------------+
| $ cht.sh go/f<tab><tab>| | $ cht.sh --shell       | | $ cht.sh go zip lists  |
| go/for   go/func       | | cht.sh> help           | | Ask any question using |
| $ cht.sh go/for        | | ...                    | | cht.sh or curl cht.sh: |
| ...                    | |                        | | /go/zip+lists          |
|                        | |                        | | (use /,+ when curling) |
|                        | |                        | |                        |
+---- TAB-completion ----+ +-- interactive shell ---+ +- programming questions-+
+------------------------+ +------------------------+ +------------------------+
| $ curl cht.sh/:help    | | $ vim prg.py           | | $ time curl cht.sh/    |
| see /:help and /:intro | | ...                    | | ...                    |
| for usage information  | | zip lists _            | | real    0m0.075s       |
| and README.md on GitHub| | <leader>KK             | |                        |
| for the details        | |             *awesome*  | |                        |
|            *start here*| |                        | |                        |
+--- self-documented ----+ +- queries from editor! -+ +---- instant answers ---+

[Follow @igor_chubin for updates][github.com/chubin/cheat.sh]
$
```

Basics

Commands that help you to navigate through the system:

- `pwd` : Within terminal a user has a current state, the directory in which the user is. This command outputs the full path to this directory.
- `whoami` : Each user in Linux has set of rights on read, write and execute files. This command will output the name of the user logged in this terminal instance.
- `cd` : Let you change your current directory.
- `pushd`, `popd` : Often you are in a situation when it is needed to change directory temporarily then return to previous location. These command create stack of directories, you can traverse back to the directory you've left the most recent
- `ls` : Lists contents of the directory
- `du` : Outputs disk usage of contents of the directory
- `clear` : Clears terminal from everything written in it

Basics

Upper level work with files:

- `mkdir` : Creates empty folder
- `touch` : Creates empty file
- `cp` , `mv` : The first one copies and the second one moves file or directory to a specific location. Note that renaming in Linux is executed with `mv` command also (`mv old_name new_name`)
- `rm` : Deletes file or directory
- `find` : Helps you to find file in filesystem

Globing

Allows you to create masks for file names and directories

1) `*` - any number of arbitrary characters. Example: `head *.txt`

2) `?` - one arbitrary character. Example: `ls 0?.txt`

3) `{aa,ab,cd,ef}` - one character from the specified group.

Example: `rm split_{a,b,c,d}.txt`

4) `[0-9]` - one character from the specified range.

Example: `gzip logs-2017-10-2[0-5].txt`

(`gzip` is a program for creating archives)

Work with file contents

- `head`, `tail` : Print the beginning or end of the file or stream - `head` and `tail`, respectively. A useful option is `-n`, which specifies how many rows to output. By default, 10.
- `grep` (is your friend!) :

  The `grep` command is used to filter the input stream. `grep ERROR very_important_log.txt`. Useful options:
  - `-i` - ignore case
  - `-v` - inverted filtering
  - `-E` - extended regular expressions
  - `-A 2` - output two lines before the matching template
  - `- B 4` - output four lines after the matched template
  - `-C 3` - context 3 lines before 3 after
  - '--color` - help your eyes and highlight the matching pattern

Work with file contents

- `uniq` --- returns unique elements from the input stream. But not quite. uniq replaces a group of consecutive unique values with a single value. Let's say we have a file `uniq_test_1.txt` with contents '1 1 1 1 2 2 2' and `uniq_test_2.txt` with the content ' 1 1 2 2 1 1`.

`cat uniq_test_1.txt | tr "" "\n" | uniq | tr "\n" ""` will output `1 2`

`cat uniq_test_2.txt | tr "" "\n" | uniq | tr "\n" ""` will output `1 2 1`

Useful options:

`-c` - counts the number of each element `- u` - outputs only unique elements `-d` - outputs only duplicates

Work with file contents

- `sort`

Sorts the input data stream. Useful options:

`-k1, 3` - sort by key starting with the 1st field and ending with the 3rd field

- `split`

Splits the source file into smaller files with the specified parameters. It may be useful for parallel processing. Example:

`split -a 4 -l 100 -d my_big_file.txt split_part_` - splits the file my_big_file.txt for files of 100 lines, it will use 4 characters under the suffix, the files will be called split_part_0000, split_part_0001. The-d option is necessary for suffixes to be formed from numbers, by default Latin letters

Work with file contents

- `tr`

This utility is needed for converting text in the input stream

`tr "[:lower:]" "[:upper:]"` - replaces all lowercase letters with the corresponding uppercase letters

`tr """\n"` - replaces spaces with line feeds

- `wc`

Utility for counting characters, words, and strings. Example `' wc -l my_big_file.txt`

Pipes and Standard Streams

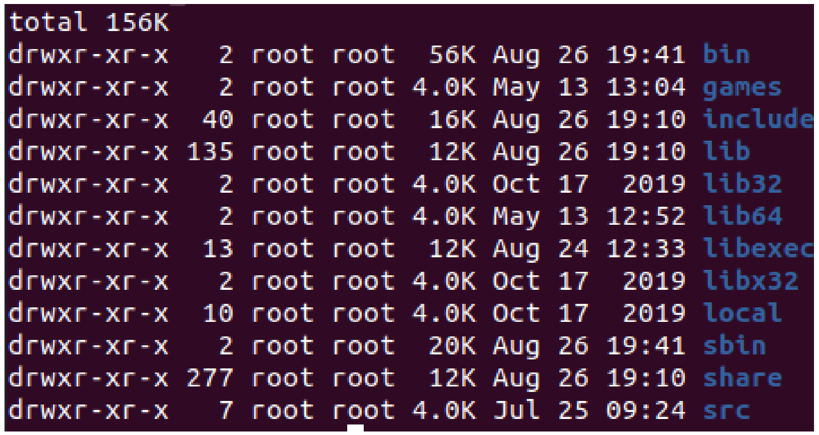
Data streams can be redirected using a special character - " | " (pipe). Then the stdout of one command becomes the stdin of the next command. Example:

```
cat my_file.txt | head | grep awesome_pattern | awk '{print $NF}' > result.txt
```

Standard streams

In a terminal there are exist three standart streams: stdin, stdout, stderr. In general case program takes input from stdin and outputs results in stdout, if there are some additional warnings or errors it outputs that information in stderr.

`ls -lh` in `/usr` dir will output something like this:



```
total 156K
drwxr-xr-x   2 root root  56K Aug 26 19:41 bin
drwxr-xr-x   2 root root 4.0K May 13 13:04 games
drwxr-xr-x  40 root root  16K Aug 26 19:10 include
drwxr-xr-x 135 root root  12K Aug 26 19:10 lib
drwxr-xr-x   2 root root 4.0K Oct 17  2019 lib32
drwxr-xr-x   2 root root 4.0K May 13 12:52 lib64
drwxr-xr-x  13 root root  12K Aug 24 12:33 libexec
drwxr-xr-x   2 root root 4.0K Oct 17  2019 libx32
drwxr-xr-x  10 root root 4.0K Oct 17  2019 local
drwxr-xr-x   2 root root  20K Aug 26 19:41 sbin
drwxr-xr-x 277 root root  12K Aug 26 19:10 share
drwxr-xr-x   7 root root 4.0K Jul 25 09:24 src
```

Conclusion

- What is Linux, how can we benefit from using it
- What is the Shell, how it helps us interact with computer
- Brief introduction to Shell commands