Bibliographie:

- The Linux Command line: http://linuxcommand.org/tlcl.php
- Data Science at the Command Line: Chez O'Reilly http://datascienceatthecommandline.com/

Pourquoi la ligne de commande ?

- pas de souris, pas de fenêtre !
- environnement ultra stable: pas changé depuis plus de 30 ans !
- principes de bases : (Doug MacIlroy 1972)
  - des programmes qui font une seule chose et qui le font bien
  - des programmes qui peuvent communiquer entre eux
  - des programmes qui manipulent du texte (le langage universel)
- CLI agile et interactive: modèle read-eval-print (REPL) versus modèle edit-compile-run-debug
- Proche du système de fichier, donc proche des données → rapidité d'exécution
- Super outil pour les flemmards → automatiser les tâches répétitives
- S'intègre magnifiquement avec d'autres technologies: C/C++, python, perl, R, ruby, etc.
- On peut créer très facilement ses proches outils (miniscripts, etc.)
- autocompletion

Files

- pushd / popd
  - pushd path_to_dir -> go to directory but remember current directory
  - popd -> go back to last remembered directory
- tree: get list contents of directories
  - tree -d : list directories only
- disk usage
  - du -h: compute size of directories
  - du -a: all files, not just directories
  - du --max-depth=1 : display only the size of the directories and not the subdirectory
- manipulate files
  - touch
  - rmdir: remove empty directories
- locate : quickly search for the location of a specific file
  - locate myfile
- synchronize with rsync
  - can sync with remote server
  - http://www.thegeekstuff.com/2010/09/rsync-command-examples/
  - rsnapshot → full backup of OS

Monitoring

- Processes
  - ps -ef / ps -aux: list of currently running processes
  - ps -u username → processes that belongs to a specific username
  - get the unique process ID (PID) of a given application (for killing it) and kill it

$ ps -aux | grep 'applicationname'
$ kill PIDnumber
    - ps –forest : display with a hierarchy
    - kill -9 PIDnumber → force kill
- Memory leak
    - ps -aux –sort pmem → sort by increasing usage of memory
    - free → display the free, used, swap memory available in the system
- top / htop
    - navigate with ">" or "<" to sort by different columns
    - q to quit
- df: displays the file system disk space usage
    - df -h : human readable

Remote

- Wakeonlan (wol) : allows to switch ON remote servers without physical access.
    - sends magic packets to wake-on-LAN enabled ethernet adapters and motherboards to switch on remote computers.
    - for shutdowns by mistake; server that don't need to be up 24x7
    - requisite:
        - know the MAC address
        - NIC (carte réseau) must support wakeonlan
          To check, type use the command
          $ ethtool eth0
            - If NO: → Cannot get wake-on-lan settings: Operation not permitted
            - If YES → Supports Wake-on: pumbg
        - enable the wakeonlan feature before shutdown
        - root access
    - http://www.thegeekstuff.com/2008/11/wol-wakeonlan-guide-remotely-turn-on-servers-without-physical-access/

Most useful tools

- grep: allows to search for a given string
    - grep "mystring" myfile
    - grep -r "mystring" * → search recursively in all files
- find: find files
    - find all txt files recursively from current directory
      find -type f -name '*.txt'
    - find -type f -name '*.txt' -exec mycommand {} \;
    - find file of size higher than 1 Go
      find -size +1G
    - find all empty files and remove them
      find -empty -exec rm {} \;
- wc: wordcount
    - wc -l : count number of lines

    - count number of file containing "mystring"
      find -type f -name '*mystring*' | wc -l
- rename: rename multiple files
    - change a file extension:
      rename 's/\.txt$/\.csv/' *.txt
    - rename 's/([0-9]{4})([0-9]{2})([0-9]{2})/$1-$2-$3/'
    - transform YYYYMMDD date to ISO format:
      find -name '*.pdf' -exec rename 's/([0-9]{4})([0-9]{2})([0-9]{2})/$1-$2-$3/' {} \;
    - rename -n → do no perform operations but show how file would have been renamed
- xargs: execute command line from standard input
    - use grep, find, awk to make a list of arguments (filenames, etc.), then pipe ("|") it to xargs
    - Warning: if there are too many arguments, it won't work. In that case use
      find /path -type f -name "mystring" -print0 | xargs -0 mycommand
      instead of
      find /path -type f -name "mystring" | xargs mycommand
    - Warning2: xargs deals badly with special characters (such as space, ' and ")
    - Search all jpg images in the home directory and archive it.
      find ~ -name *.jpg -type f -print | xargs tar -cvzf images.tar.gz
    - Copy all images to external hard-drive
      ls *.jpg | xargs -n1 -i cp {} /external-hard-drive/directory
- gnu parallel
    - works (by design) like xargs
    - uses all available cpu to run jobs in parallel (xargs run jobs in parallel but does not manage them according to number of cpu)
    - gnuparallel keeps the order of the output (not xargs, so if running jobs in parallel using xargs the output of the second job cannot be postponed till the first job is done)
    - can run job on remote machines (not xargs)
    - For a list of differences with GNU parallel alternatives, see:
      http://www.gnu.org/software/parallel/man.html#DIFFERENCES-BETWEEN-xargs-AND-GNU-Parallel

Manipulate files

- cat → display file in sdtout
- head -n → n first lines
- tail -n → n last lines
  tail -f → display the end of the file (
- sed: stream editor for filtering and transforming text
    - Add a prefix to each line
      sed -e 's/^/myprefix/'
    - Insert a line at line number 4
      sed -i '4i mynewline' myfile
    - convert from DOS format to UNIX (get rid of characters such as "\r\n")
        - dos2unix myDOSfile myUNIXfile
        - sed 's/.$//' myDOSfile
        - sed -ie 's/[ctrl+v][ctrl+m]//' myDOSfile
          (-i for inplace; -e for expression (regex))

awk

- Named after its authors Alfred Aho, Peter Weinberger, and Brian Kernighan (Bell Labs)
- Appeared in 1977
- Excellent for working on csv
- http://www.grymoire.com/Unix/Awk.html
- Some simple examples:
  - print last column
    awk '{print $NF}' myfile
  - print first column for a
    awk -F "|" '{print $1}' myfile
  - print line number if number of field is not 12
    awk 'NF!=12{print NR}' myfile
  - convert to lowercase:
    awk '{print tolower($0)}' myfile
  - print in order the distinct values of column 2 (tab separated file)
    awk -F '\t' '{print $2}' myfile | sort | uniq

csvkit

- Written in python
- tools for working with csvkit.readthedocs.org
  - csvcut (move columns and rows around)
  - csvjoin
  - csvsort
  - csvjson
  - csvlook
  - csvstacks
  - csvstats
  - sql2csv
  - many more !!!
- https://csvkit.readthedocs.org/en/0.9.0/