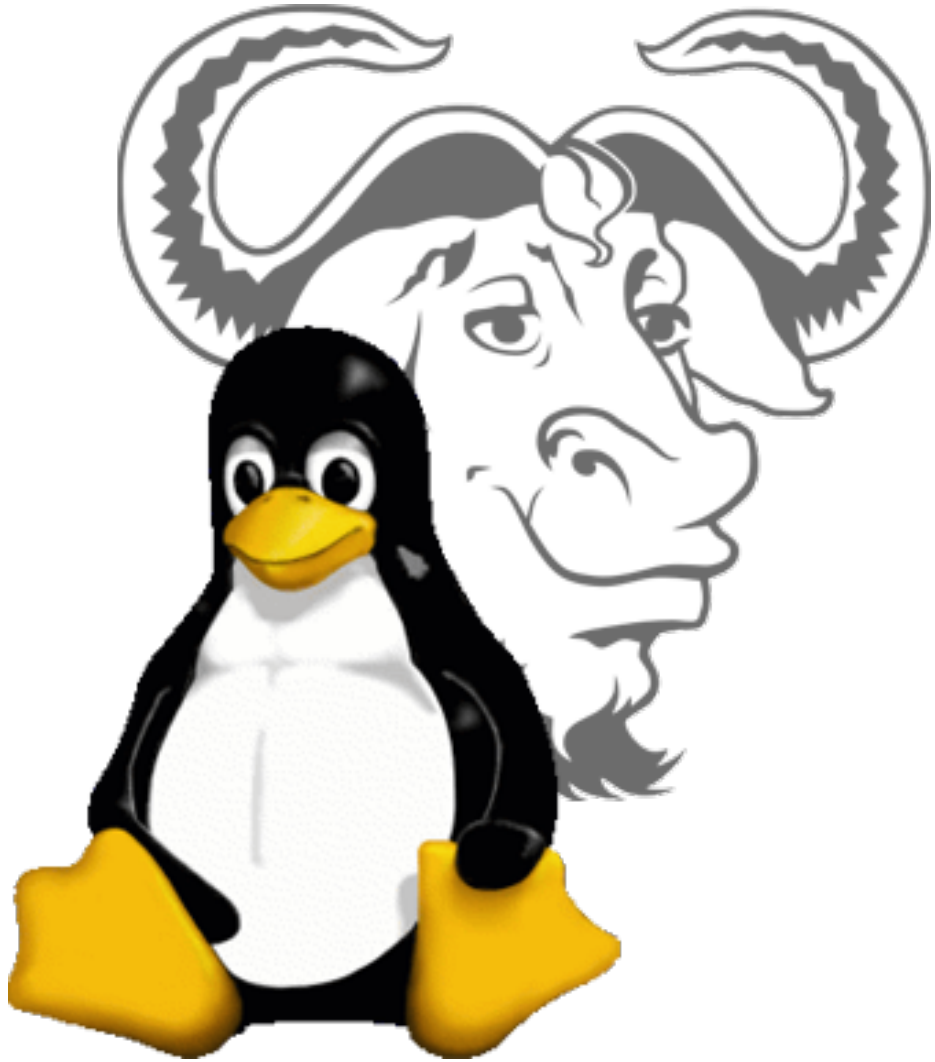
Modern C++ for Computer Vision Tutorial : Working with Linux

Ignacio Vizzo, Rodrigo Marcuzzi, Cyrill Stachniss

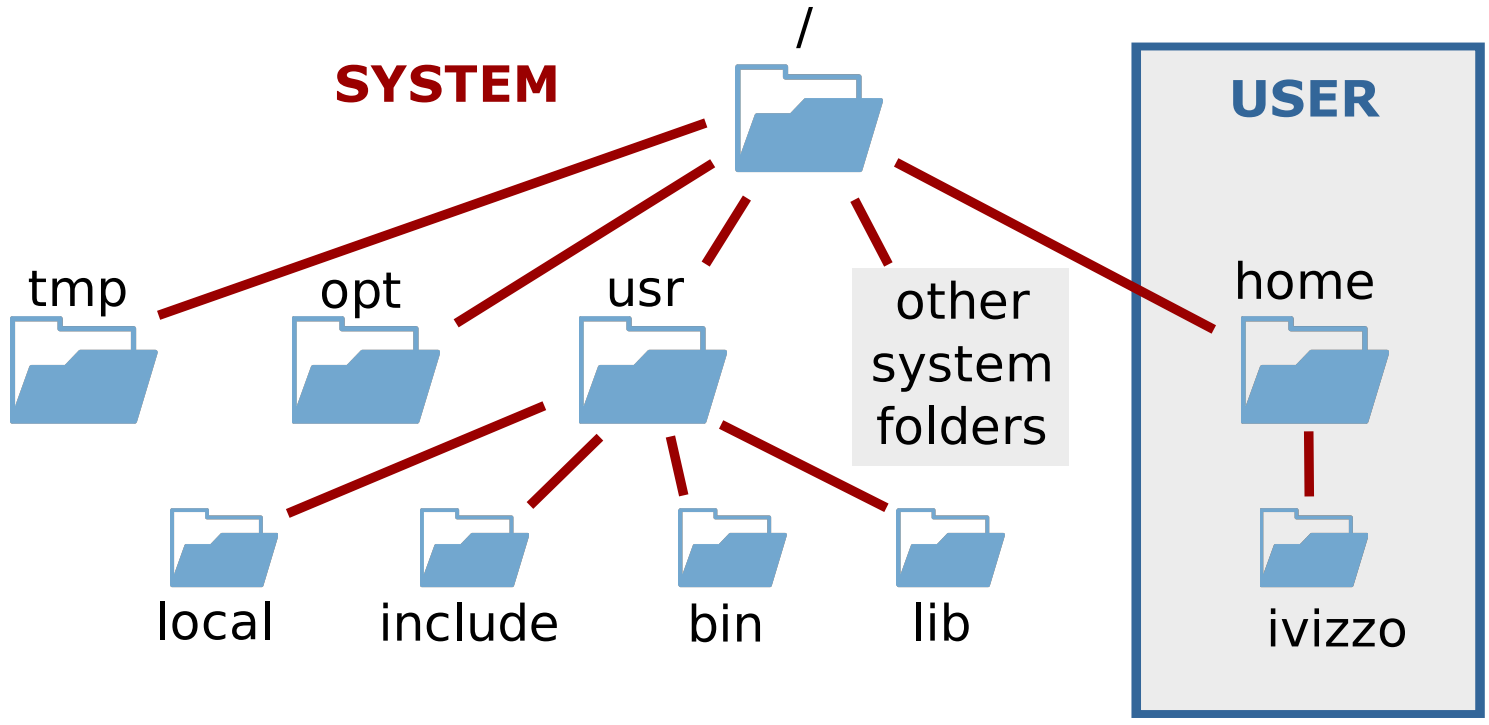


Icon taken from Wikipedia

What is GNU/Linux?

- Linux is a free **Unix-like OS**
- Linux kernel implemented by Linus Torvalds
- **Extremely popular:** Android, ChromeOS, servers, supercomputers, etc.
- Many **Linux distributions** available
- Use any distribution if you have preference
- Examples will be given in **Ubuntu**

Linux directory tree



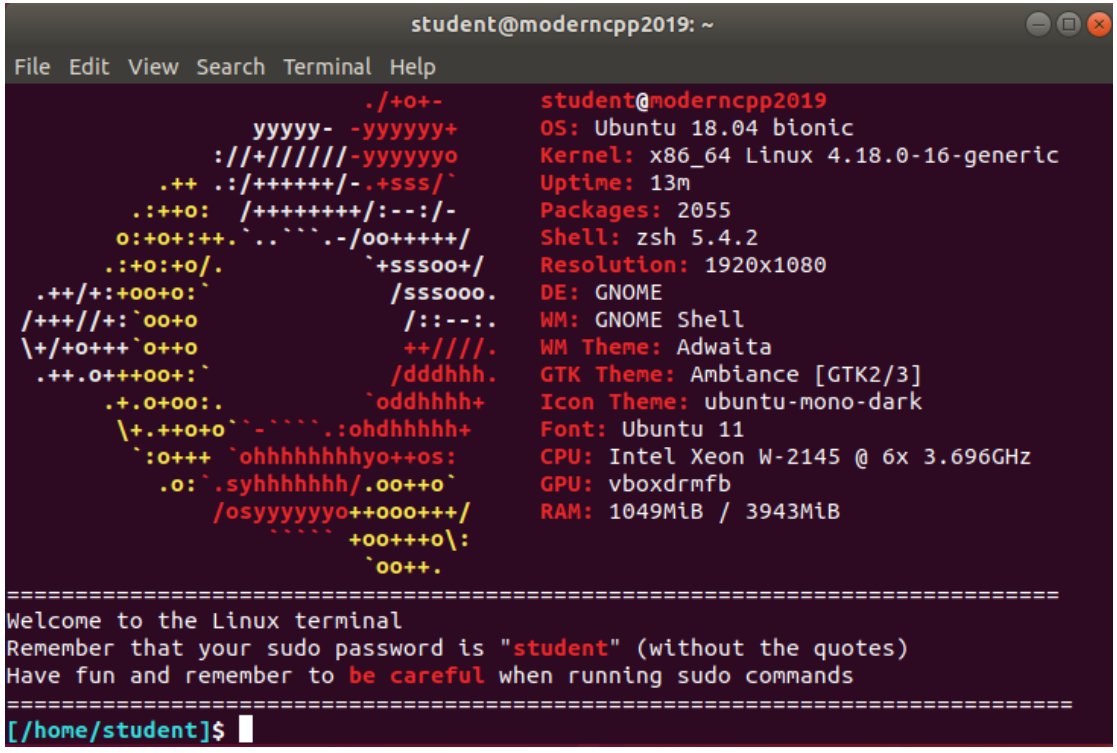
- Tree organization starting with root: `/`
- There are no volume letters, e.g. `C:`, `D:`
- User can only access his/her own folder

Understanding files and folders

- Folders end with `/` e.g. `/path/folder/`
- Everything else is files, e.g. `/path/file`
- Absolute paths start with `/`
  while all other paths are relative:
  - `/home/ivizzo/folder/` — **absolute** path to a folder
  - `/home/ivizzo/file.cpp` — **absolute** path to a file
  - `folder/file` — **relative** path to a file
- Paths are case sensitive:
  `filename` is different from `FileName`
- Extension is part of a name:
  `filename.cpp` is different from `filename.png`

Linux terminal

- Press Ctrl + Alt + T to open terminal



- Most tasks can be done faster from the terminal than from the GUI

Navigating tree from terminal

- Terminal is always in some folder
- `pwd`: **p**rint **w**orking **d**irectory
- `cd <dir>`: **c**hange **d**irectory to `<dir>`
- `ls <dir>`: **l**i**s**t contents of a directory

- Special folders:
  - `/` — root folder
  - `~` — home folder
  - `.` — current folder
  - `..` — parent folder

Structure of Linux commands

Typical structure

```
${PATH}/command [ options ] [ parameters ]
```

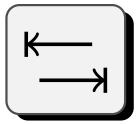
- `${PATH}/command`: obsolute or relative path to the program binary
- `[options]`: program-specific options
  e.g. `-h`, or `--help`
- `[parameters]`: program-specific parameters
  e.g. input files, etc.

Use help with Linux programs

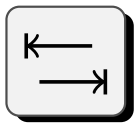
- `man <command>` — **man**ual
  exhaustive manual on program usage
- `command -h/--help`
  usually shorter help message

```
[/home/student]$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.
  -A, --show-all            equivalent to -vET
  -b, --number-nonblank     number nonempty output lines

Examples:
  cat f -  Output fs contents, then standard input.
  cat      Copy standard input to standard output.
```

Using command completion

Pressing  while typing:

- completes name of a file, folder or program
- "beeps" if current text does not match any file or folder uniquely

Pressing  **twice** shows all potential matches

Example:

```
[/home/student]$ cd D [TAB] [TAB]
Desktop/    Documents/  Downloads/
```

Files and folders

- `mkdir [-p] <foldername>` — **m**a**k**e **dir**ectory
  Create a folder `<foldername>`
  (with all parent folders `[-p]`)
- `rm [-r] <name>` — **r**e**m**ove [recursive]
  Remove file or folder `<name>`
  (With folder contents `[-r]`)
- `cp [-r] <source> <dest>` — **c**o**p**y
  Copy file or folder from `<source>` to `<dest>`
- `mv <source> <dest>` — **m**o**v**e
  Move file or folder from `<source>` to `<dest>`

Using placeholders

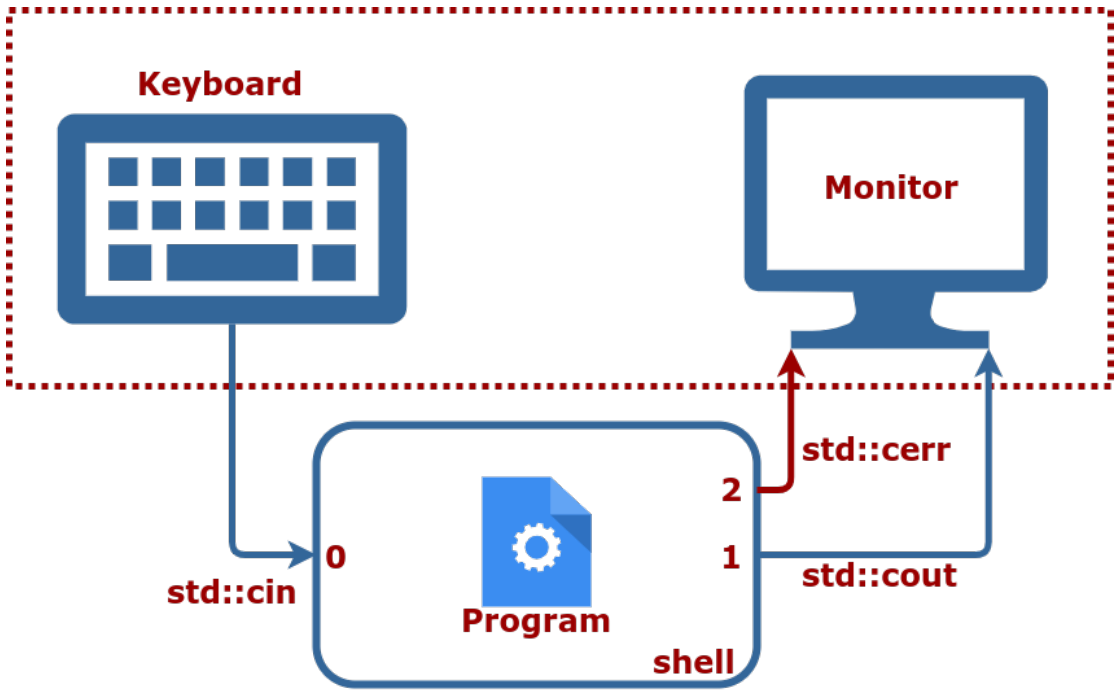
Placeholder	Meaning
`*`	Any set of characters
`?`	Any single character
`[a-f]`	Characters in `[abcdef]`
`[^a-c]`	Any character **not** in `[abc]`

Can be used with most of terminal commands: `ls`, `rm`, `mv` etc.

```
[/home/student/Examples/placeholders]$ ls
u01.tex    v01.pdf     v01.tex
u02.tex    v02.pdf     v02.tex
u03.tex    v03.pdf     v03.tex

[/home/student/Examples/placeholders]$ ls *.pdf
v01.pdf   v02.pdf   v03.pdf

[/home/student/Examples/placeholders]$ ls u*
u01.tex   u02.tex   u03.tex

[/home/student/Examples/placeholders]$ ls ?01*
u01.tex    v01.pdf   v01.tex

[/home/student/Examples/placeholders]$ ls [uv]01*
u01.tex    v01.pdf   v01.tex

[/home/student/Examples/placeholders]$ ls u0[^12].tex
u03.tex
```

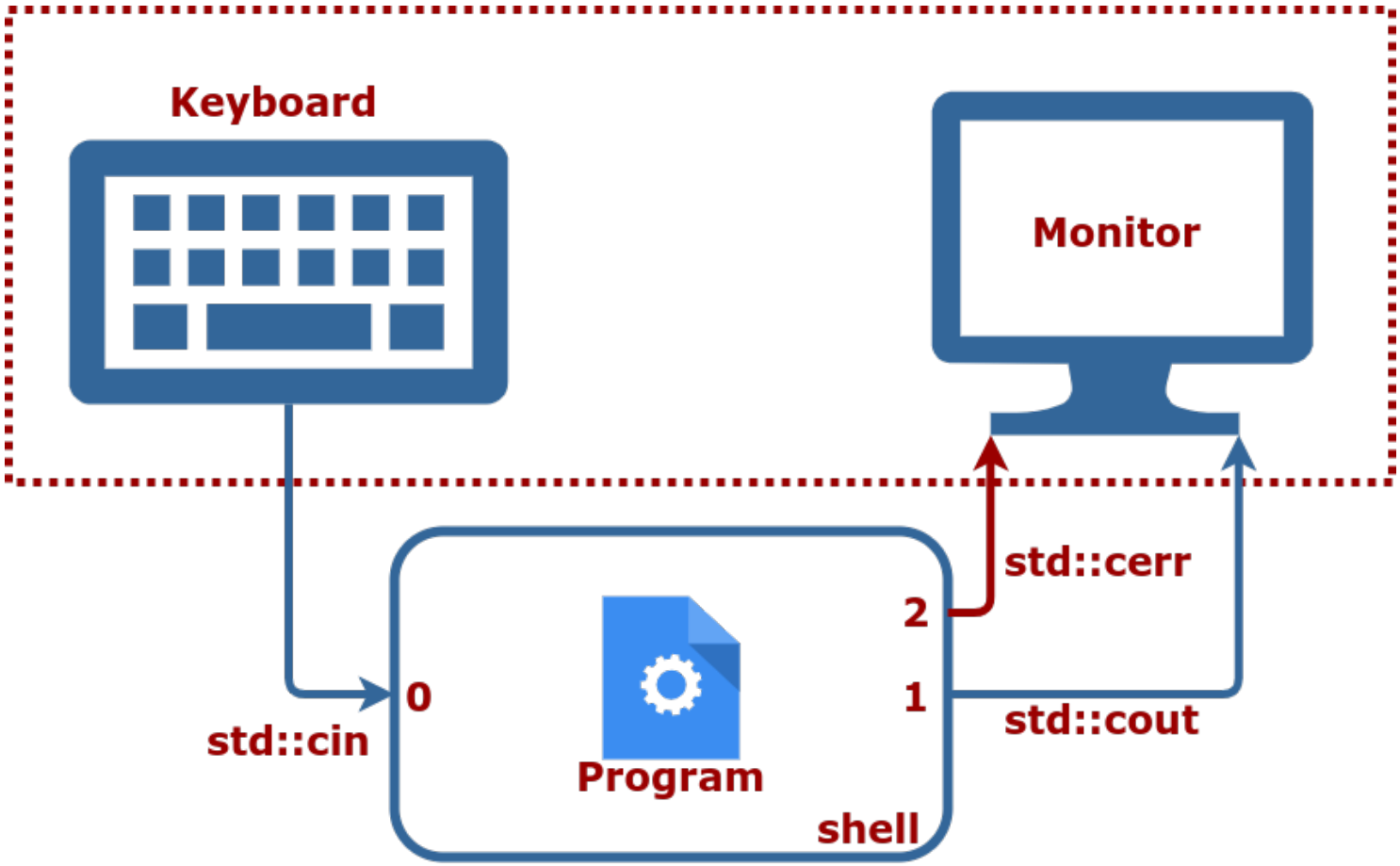
Standard input/output channels

- Single input channel:
  - `stdin`: **St**an**d**ard **in**put: channel 0
- Two output channels:
  - `stdout`: **St**an**d**ard **out**put: channel 1
  - `stderr`: **St**an**d**ard **err**or output: channel 2



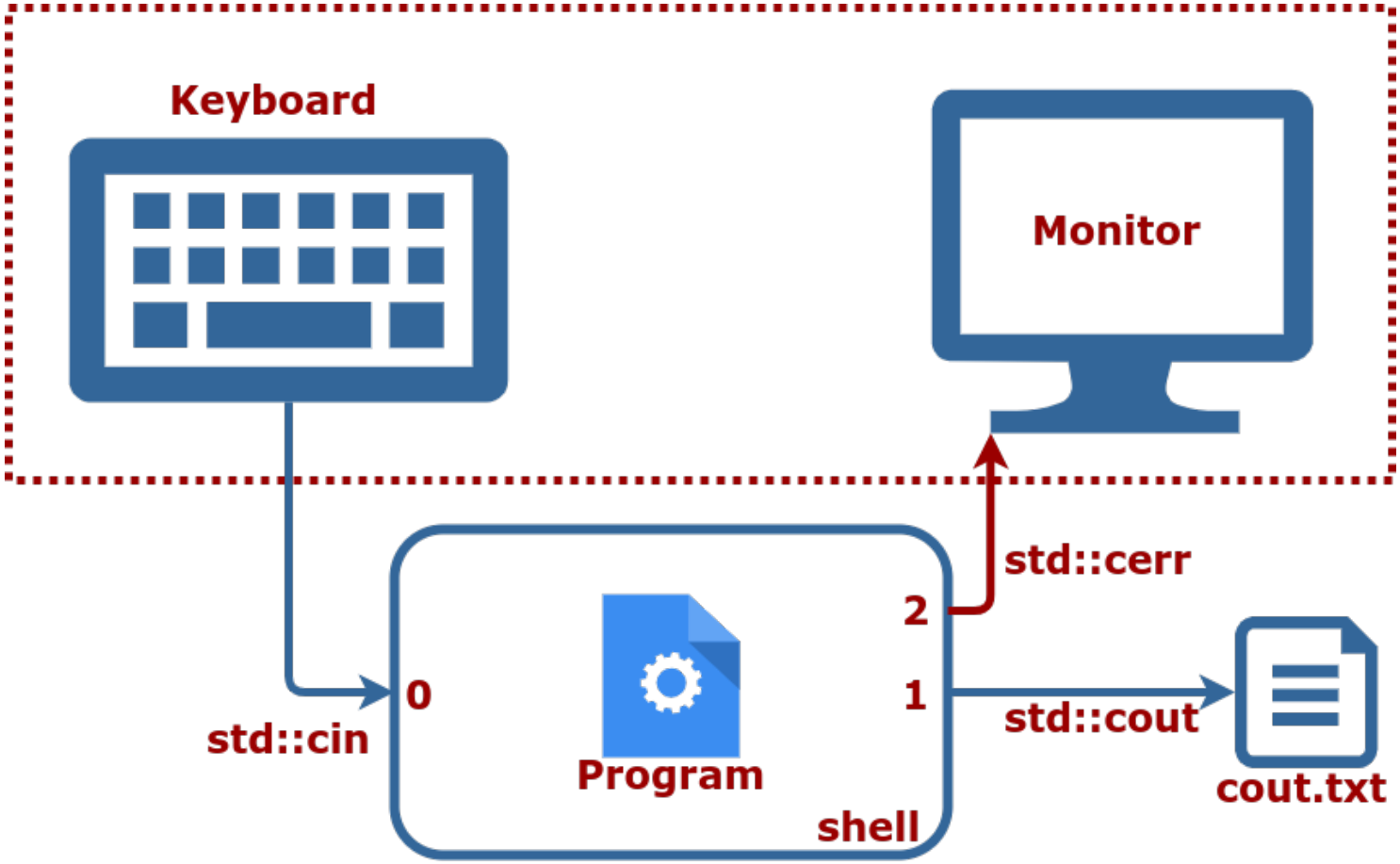
Standard input/output channels

```
$ program
```



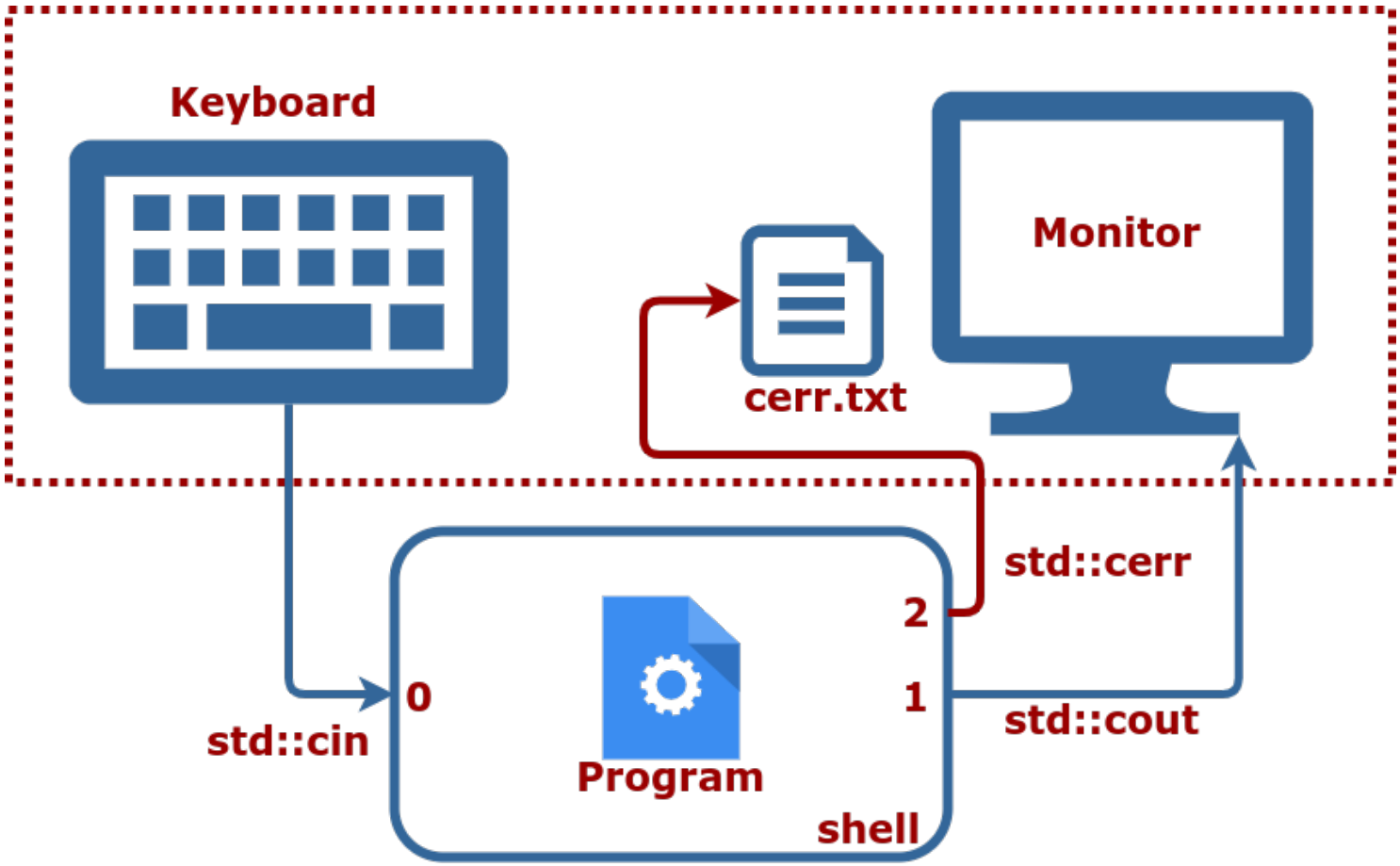
Redirecting `stdout`

```
$ program 1>cout.txt
```



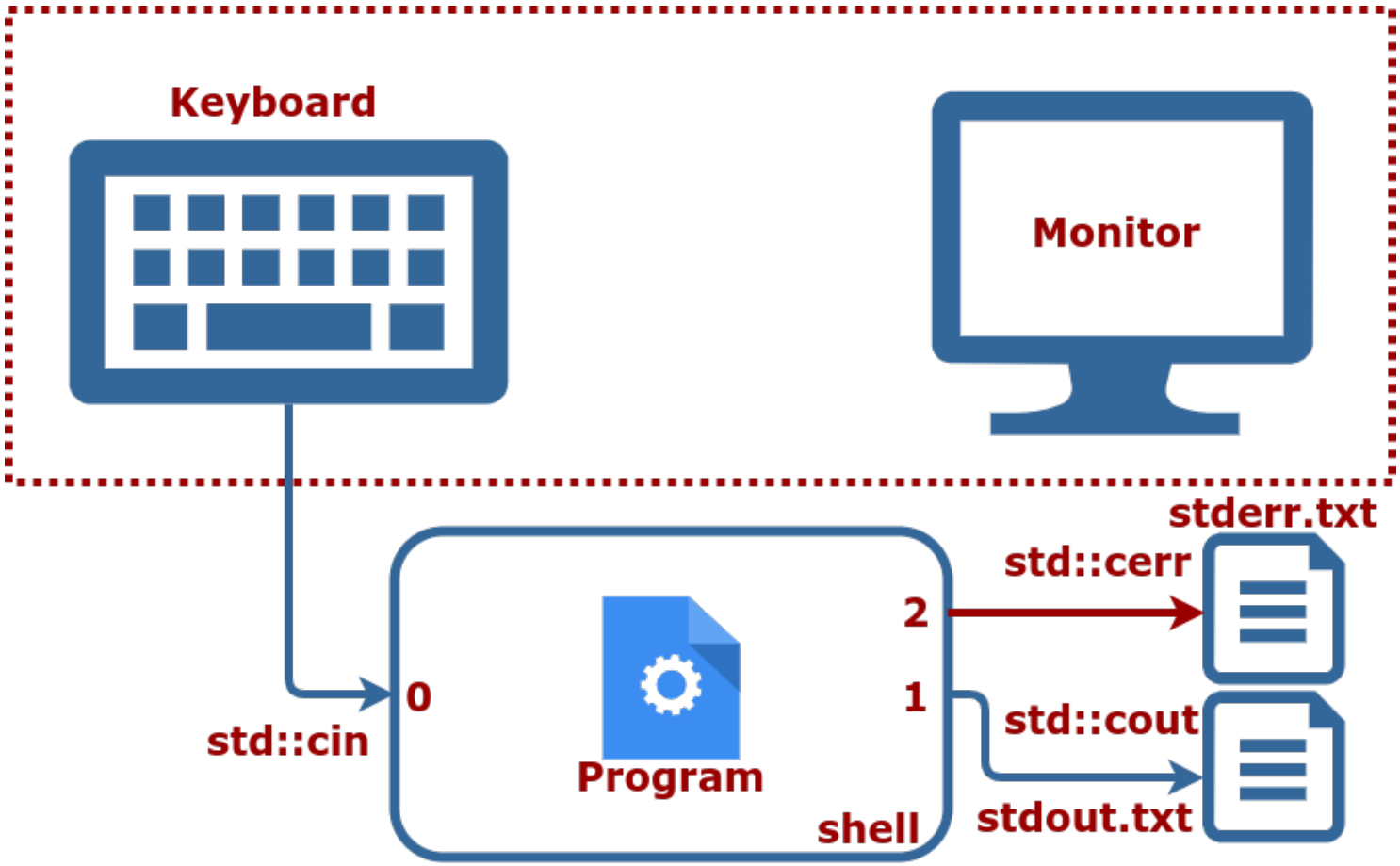
Redirecting `stderr`

```
$ program 2>cerr.txt
```



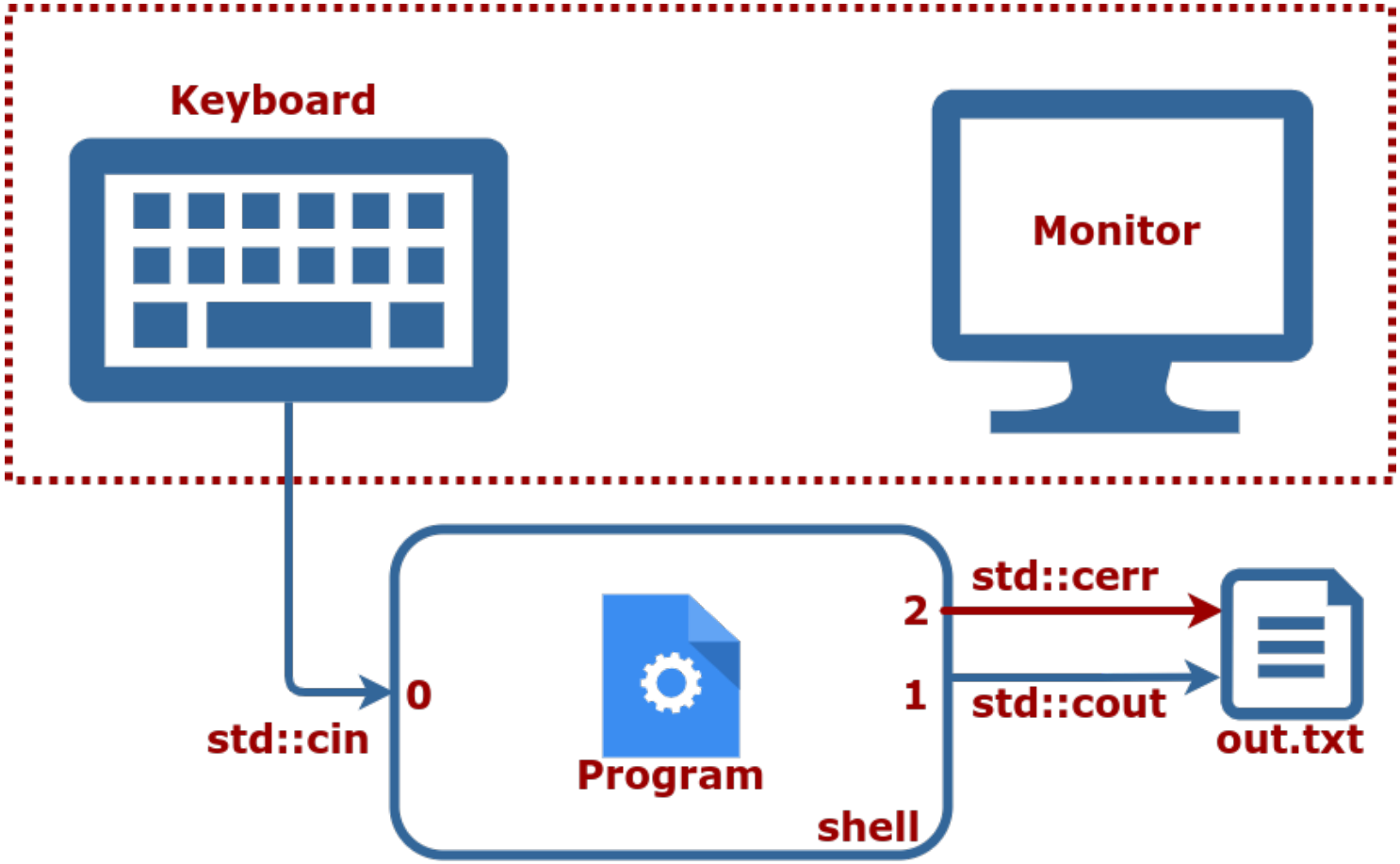
Redirect stdout and stderr

```
$ program 1>stdout.txt 2>stderr.txt
```



Redirect `stdout` and `stderr`

```
progamm 1>out.txt 2>&1
```



Working with files

- `more/less/cat <filename>`
  Print the contents of the file
  Most of the time using `cat` if enough
- `find <in-folder> -name <filename>`
  Search for file `<filename>` in folder `<in-folder>`, allows wildcards
- `locate <filename>`
  Search for file `<filename>` in the entire system!
  just remember to `sudo updatedb` often
- `grep <what> <where>`
  Search for a string `<what>` in a file `<where>`
- `ag <what> <where>`
  Search for a string `<what>` in a dir `<where>`

Chaining commands

- `command1; command2; command3`
  Calls commands one after another
- `command1 && command2 && command3`
  Same as above but fails if any of the commands returns an error code
- `command1 | command2 | command3`
  Pipe `stdout` of `command1` to `stdin` of `command2` and `stdout` of `command2` to `stdin` of `command3`
- Piping commonly used with `grep`:
  `ls | grep smth` look for `smth` in output of `ls`

Linux Command Line Pipes and Redirection

https://youtu.be/mV_8GbzwZMM

Canceling commands

- `CTRL + C`
  Cancel currently running command
- `kill -9 <pid>`
  Kill the process with id `pid`
- `killall <pname>`
  Kill all processes with name `pname`
- `htop` (`top`)
  - Shows an overview of running processes
  - Allows to kill processes by pressing k

Command history

The shell saves the history of the last executed commands

- ↑: go to the previous command
- ↓: go to the next command
- Ctrl + R `<query>`: search in history
- ! + 10: execute the 10th command
- `history`: show history

Installing software

Most of the software is available in the system repository. To install a program in Ubuntu type this into terminal:

- `sudo apt update` to update information about available packages
- `sudo apt install <program>` to install the program that you want
- Use `apt search <program>` to find all packages that provide `<program>`
- Same for any library, just with `lib` prefix

Bash tutorial

https://youtu.be/oxuRxtrO2Ag

Thank you for your attention.