

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0117 Programación Bajo Plataformas Abiertas
III ciclo 2022

Laboratorio número 1 de Github

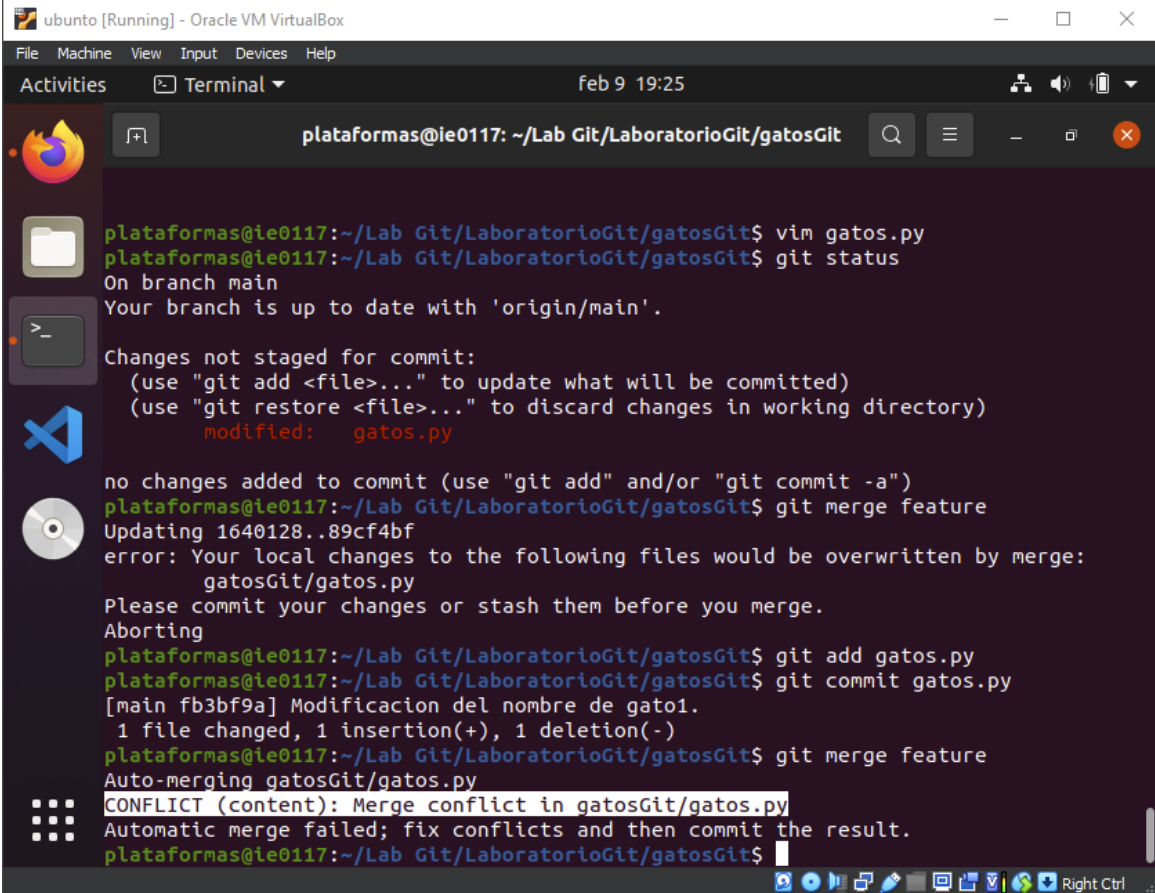
Kevin Campos Castro, B91519

Profesor: Julián Gairaud

10/01/2023

1. Reporte

5.



The screenshot shows a terminal window titled 'ubuntu [Running] - Oracle VM VirtualBox'. The terminal is running a series of git commands. The user is in a directory named 'gatosGit'. They first edit 'gatos.py' with vim, then check the status. They attempt to merge the 'feature' branch, but a conflict arises because their local changes to 'gatos.py' would be overwritten. They abort the merge, add their changes, and commit them. Then they attempt to merge 'feature' again, which results in a 'CONFLICT (content): Merge conflict in gatosGit/gatos.py'. The terminal output is as follows:

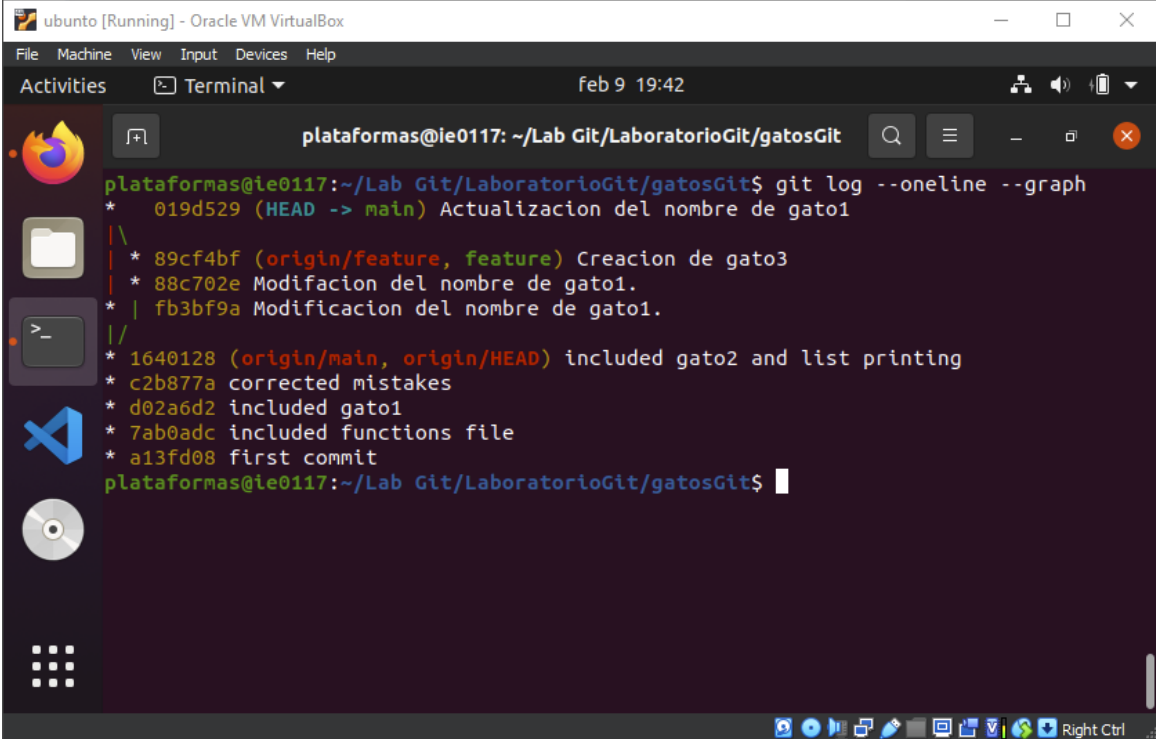
```
plataformas@ie0117: ~/Lab Git/LaboratorioGit/gatosGit
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ vim gatos.py
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   gatos.py

no changes added to commit (use "git add" and/or "git commit -a")
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git merge feature
Updating 1640128..89cf4bf
error: Your local changes to the following files would be overwritten by merge:
      gatosGit/gatos.py
Please commit your changes or stash them before you merge.
Aborting
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git add gatos.py
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git commit gatos.py
[main fb3bf9a] Modificacion del nombre de gato1.
 1 file changed, 1 insertion(+), 1 deletion(-)
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git merge feature
Auto-merging gatosGit/gatos.py
CONFLICT (content): Merge conflict in gatosGit/gatos.py
Automatic merge failed; fix conflicts and then commit the result.
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$
```

Figura 1: Uso del comando merge y el conflicto que genera.

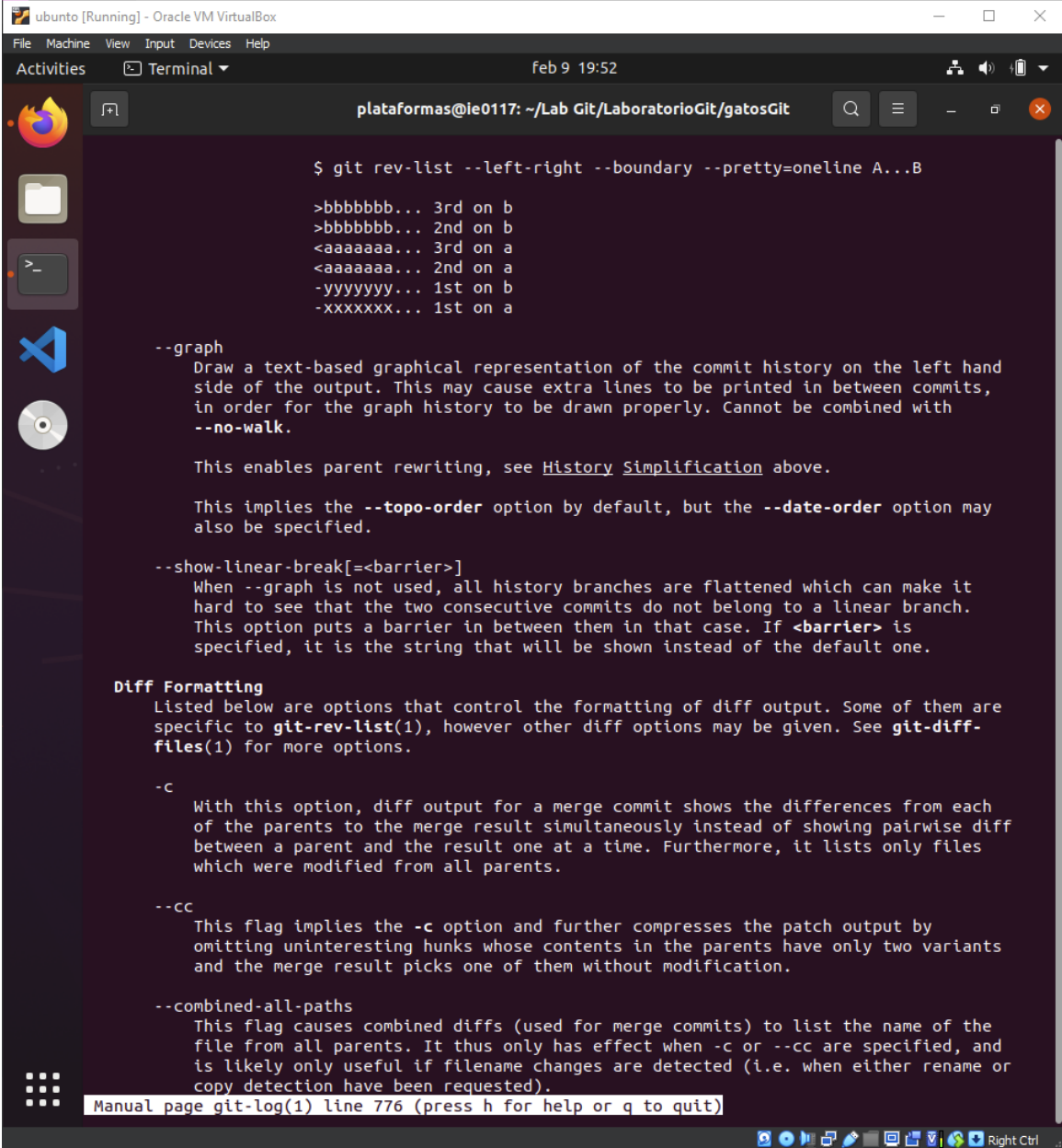
7.



```
plataformas@ie0117: ~/Lab Git/LaboratorioGit/gatosGit
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ git log --oneline --graph
* 019d529 (HEAD -> main) Actualizacion del nombre de gato1
|
| * 89cf4bf (origin/feature, feature) Creacion de gato3
| * 88c702e Modifacion del nombre de gato1.
| * fb3bf9a Modificacion del nombre de gato1.
|/
* 1640128 (origin/main, origin/HEAD) included gato2 and list printing
* c2b877a corrected mistakes
* d02a6d2 included gato1
* 7ab0adc included functions file
* a13fd08 first commit
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$
```

Figura 2: Uso del comando git log --oneline --graph.

8.
a.



```
$ git rev-list --left-right --boundary --pretty=oneline A...B
>bbbbbb... 3rd on b
>bbbbbb... 2nd on b
<aaaaaa... 3rd on a
<aaaaaa... 2nd on a
-yyy-yyyy... 1st on b
-xxxxxxx... 1st on a
```

--graph
Draw a text-based graphical representation of the commit history on the left hand side of the output. This may cause extra lines to be printed in between commits, in order for the graph history to be drawn properly. Cannot be combined with **--no-walk**.

This enables parent rewriting, see [History Simplification](#) above.

This implies the **--topo-order** option by default, but the **--date-order** option may also be specified.

--show-linear-break[=<barrier>]
When **--graph** is not used, all history branches are flattened which can make it hard to see that the two consecutive commits do not belong to a linear branch. This option puts a barrier in between them in that case. If **<barrier>** is specified, it is the string that will be shown instead of the default one.

Diff Formatting
Listed below are options that control the formatting of diff output. Some of them are specific to **git-rev-list(1)**, however other diff options may be given. See **git-diff-files(1)** for more options.

-c
With this option, diff output for a merge commit shows the differences from each of the parents to the merge result simultaneously instead of showing pairwise diff between a parent and the result one at a time. Furthermore, it lists only files which were modified from all parents.

--cc
This flag implies the **-c** option and further compresses the patch output by omitting uninteresting hunks whose contents in the parents have only two variants and the merge result picks one of them without modification.

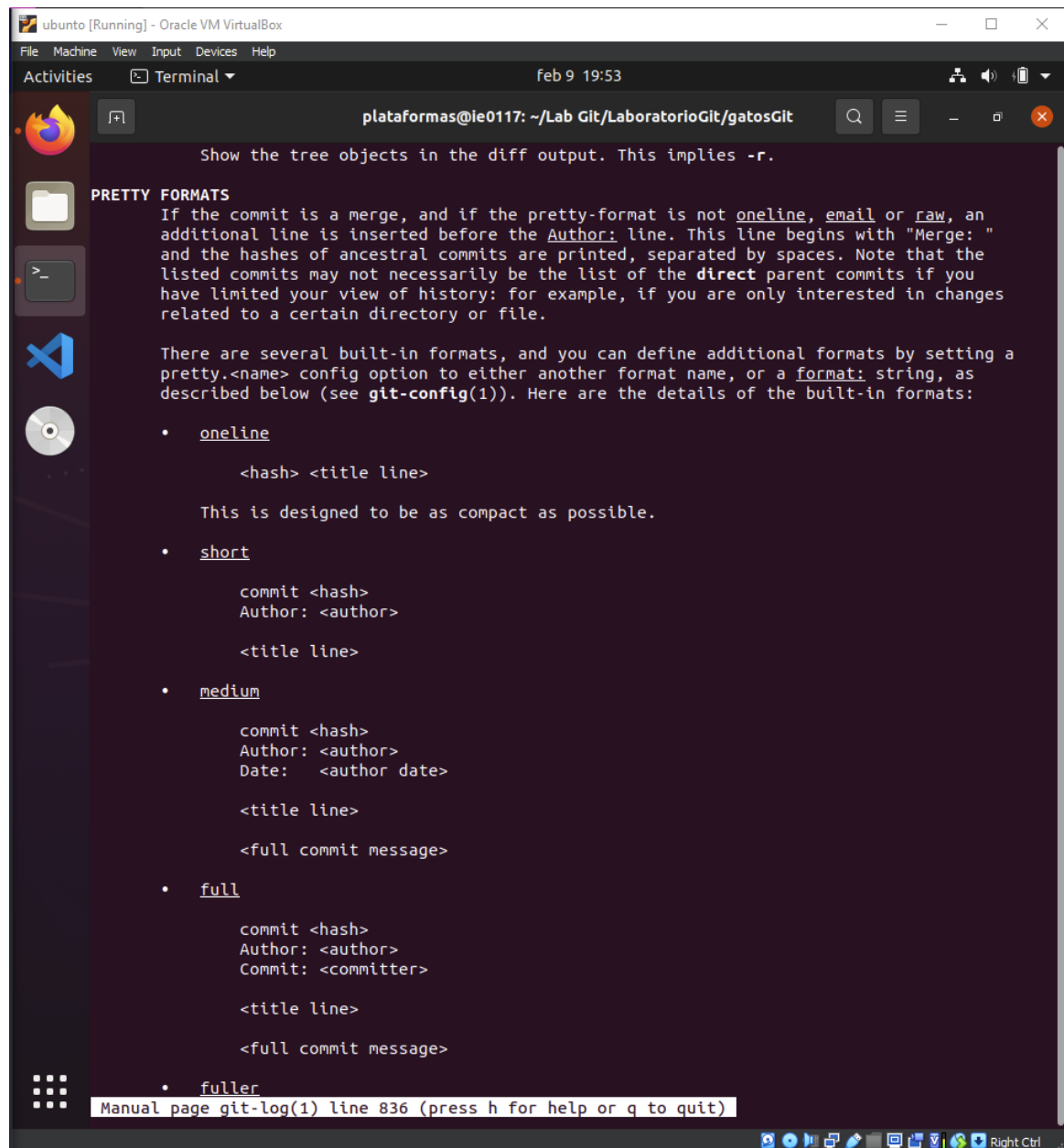
--combined-all-paths
This flag causes combined diffs (used for merge commits) to list the name of the file from all parents. It thus only has effect when **-c** or **--cc** are specified, and is likely only useful if filename changes are detected (i.e. when either rename or copy detection have been requested).

Manual page git-log(1) line 776 (press h for help or q to quit)

Figura 3: Uso del comando `git -help log (graph)`.

b., c.

En la siguiente figura se adjunta que significa el comando online y su diseño (por eso solo 2 screenshots en lugar de 3).



```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal feb 9 19:53
plataformas@ie0117: ~/Lab Git/LaboratorioGit/gatosGit

Show the tree objects in the diff output. This implies -r.

PRETTY FORMATS
If the commit is a merge, and if the pretty-format is not oneline, email or raw, an
additional line is inserted before the Author: line. This line begins with "Merge: "
and the hashes of ancestral commits are printed, separated by spaces. Note that the
listed commits may not necessarily be the list of the direct parent commits if you
have limited your view of history: for example, if you are only interested in changes
related to a certain directory or file.

There are several built-in formats, and you can define additional formats by setting a
pretty.<name> config option to either another format name, or a format: string, as
described below (see git-config(1)). Here are the details of the built-in formats:

• oneline

    <hash> <title line>

    This is designed to be as compact as possible.

• short

    commit <hash>
    Author: <author>

    <title line>

• medium

    commit <hash>
    Author: <author>
    Date: <author date>

    <title line>

    <full commit message>

• full

    commit <hash>
    Author: <author>
    Commit: <committer>

    <title line>

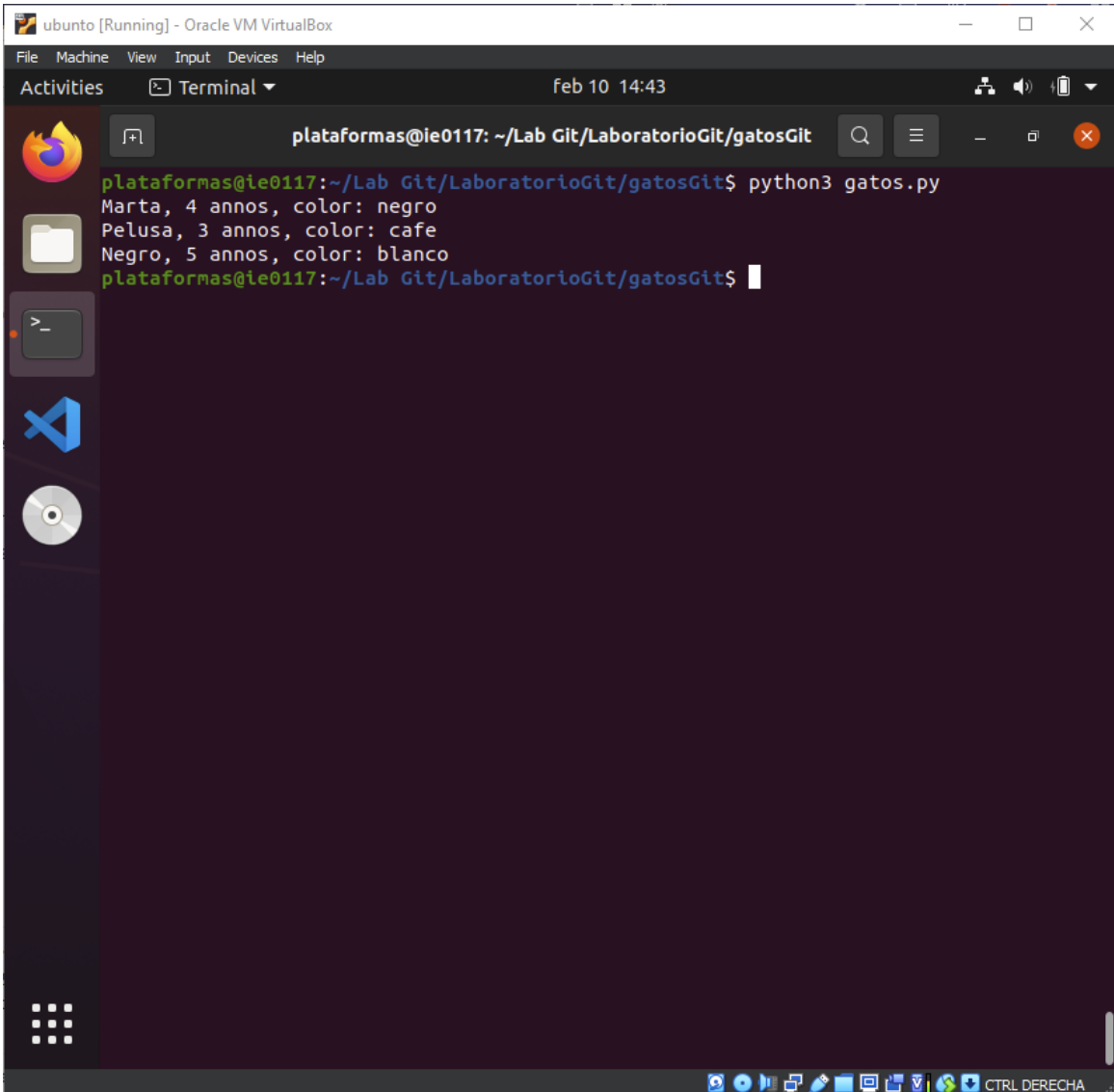
    <full commit message>

• fuller

Manual page git-log(1) line 836 (press h for help or q to quit)
```

Figura 4: Uso del comando git --help log (oneline) y su diseño.

9.



```
plataformas@ie0117: ~/Lab Git/LaboratorioGit/gatosGit
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$ python3 gatos.py
Marta, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Negro, 5 annos, color: blanco
plataformas@ie0117:~/Lab Git/LaboratorioGit/gatosGit$
```

Figura 5: Resultado de correr gatos.py.