

FRAMES, WINDOWS E BUFFERS (IN EMACS)

LEONARDO TAMIANO

Created: 2023-12-31 Sun 11:51

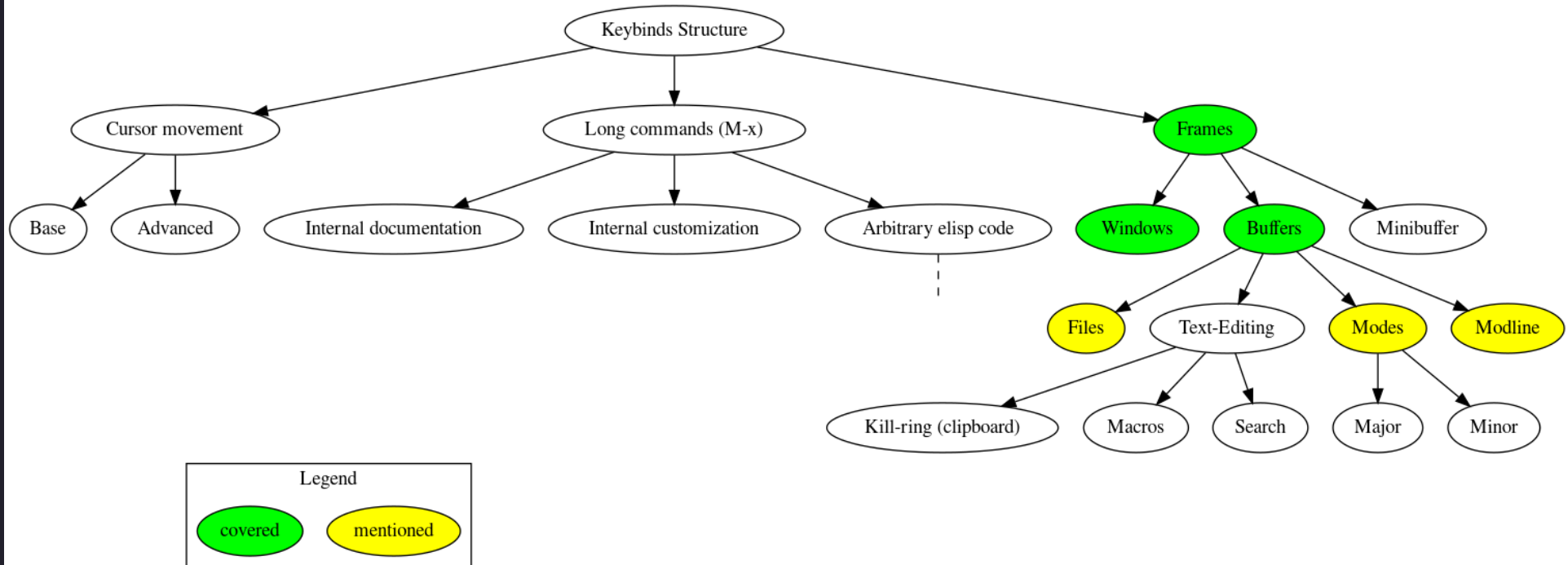
TABLE OF CONTENTS

- Overview
- Frames
- Windows
- Buffers

OVERVIEW

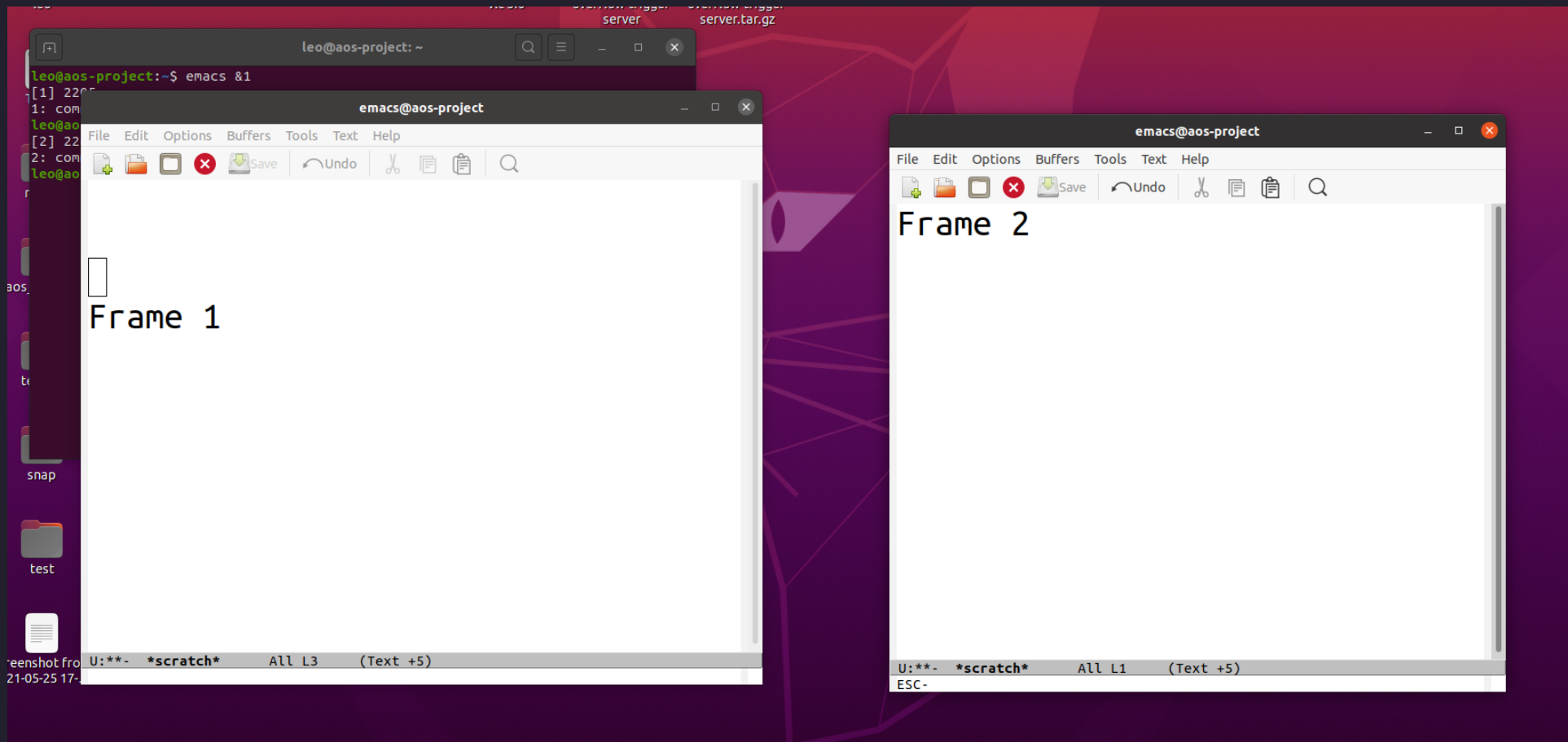
Utilizzando lo schema introdotto possiamo inquadrare i concetti che affronteremo in questo video.

Emacs Baseline Concepts



FRAMES

Nel linguaggio di Emacs, per frame si intende la **finestra**
GUI che contiene che contiene il programma in un
ambiente grafico come X11.



Due frames aperti.

È possibile modificare le dimensioni del primo frame aperto con il seguente codice LISP

```
(setq initial-frame-alist '((top . 50)  
                             (left . 150)  
                             (width . 70)  
                             (height . 20)))
```

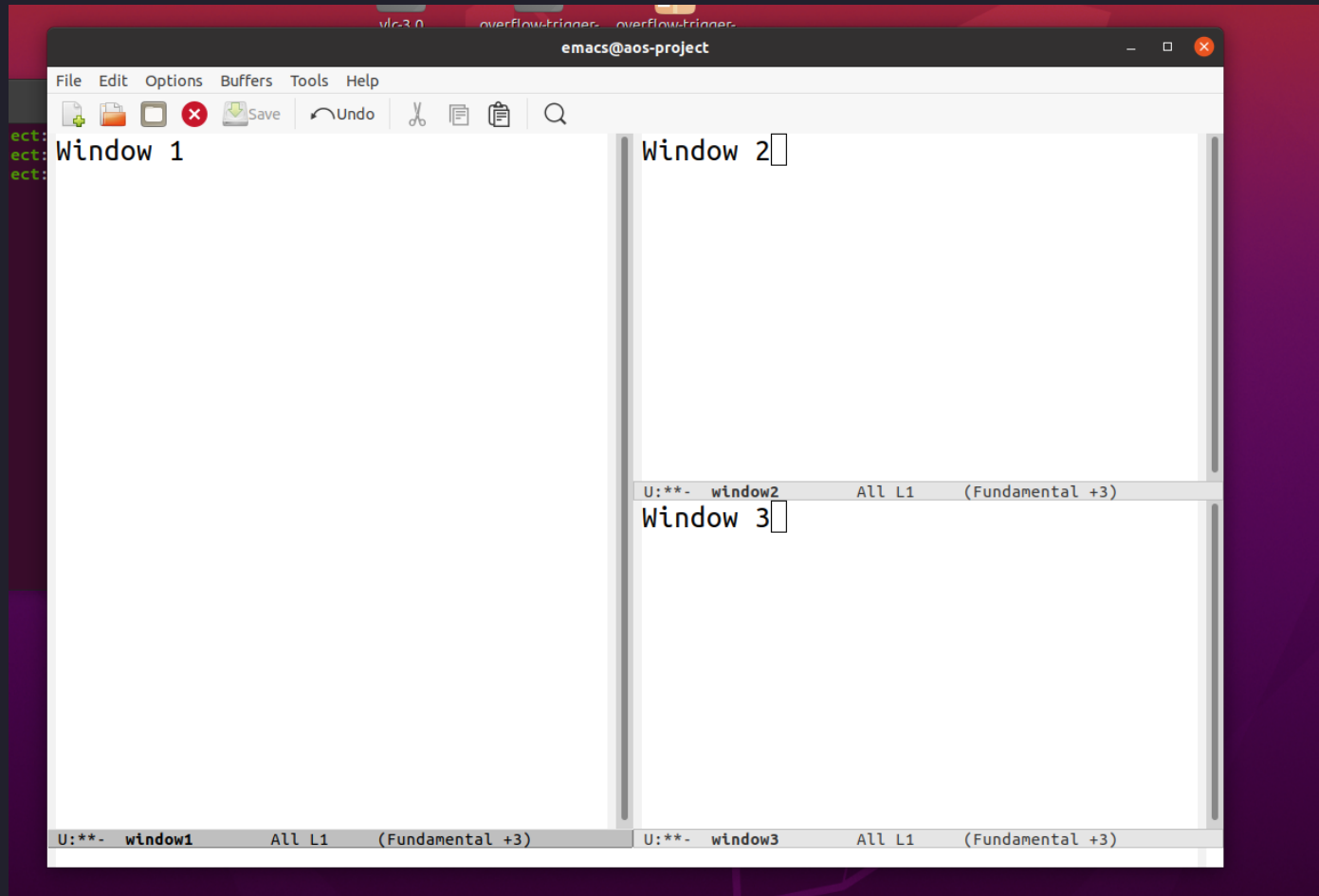
Alcuni keybinds per gestire i frame sono i seguenti

Keybind	Comando
C-x 5 2	make-frame
C-x 5 o	other-frame

WINDOWS

Per Emacs una **window** (finestra) è una porzione dello schermo visibile.

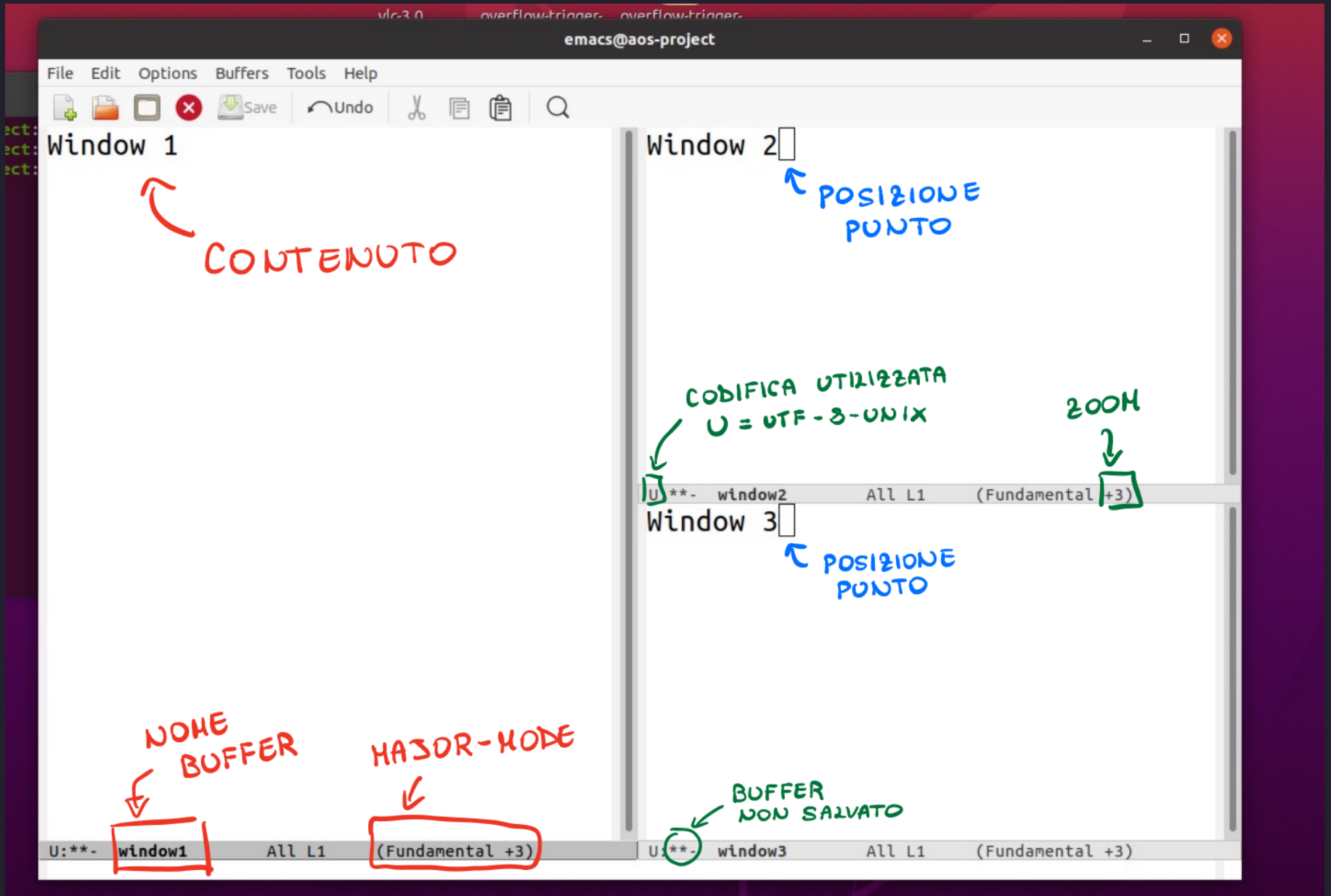
Un frame di Emacs può essere diviso in più finestre (windows).



Frame diviso in tre windows.

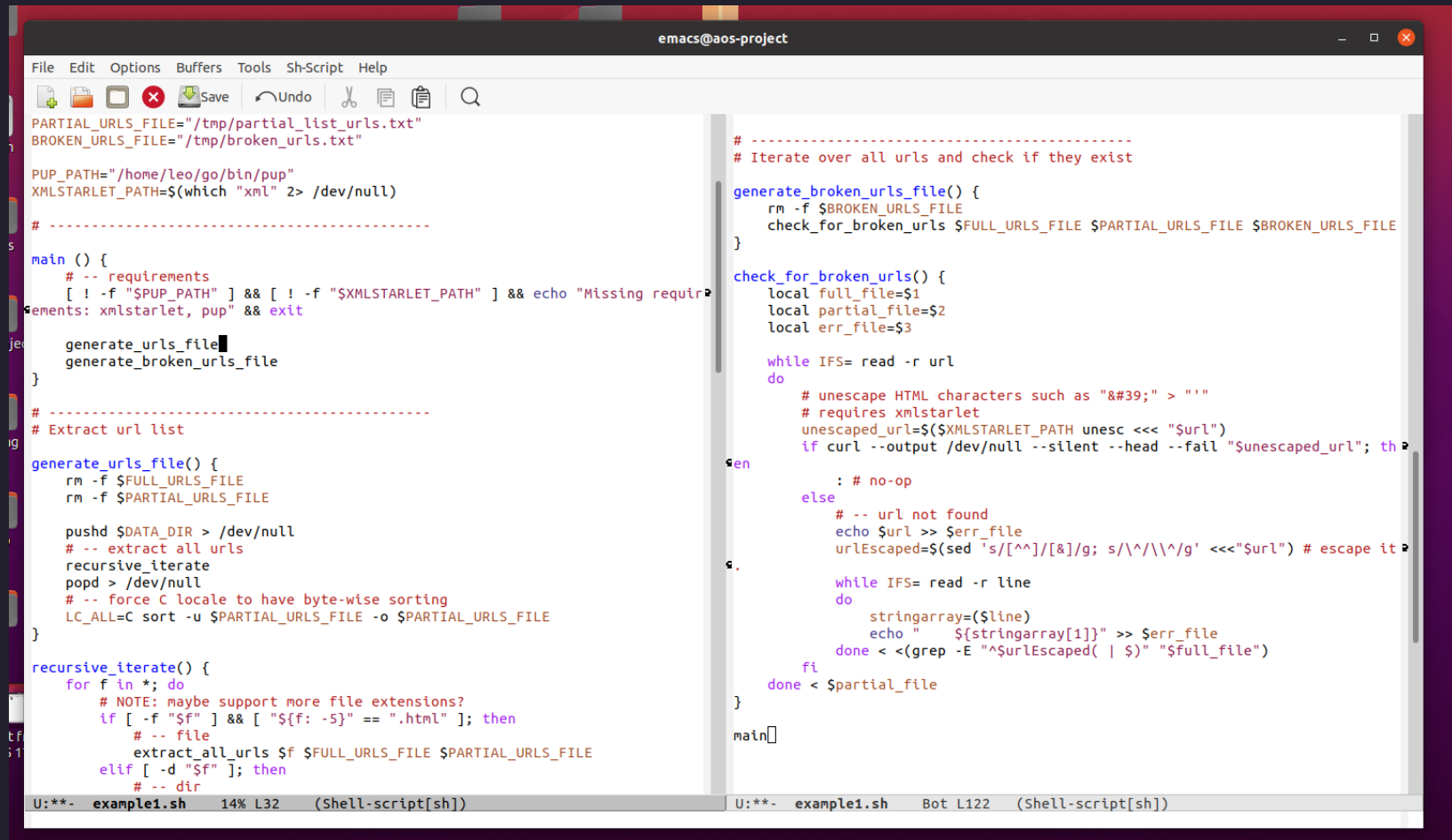
Ciascuna finestra mostra una porzione di un **buffer**.
Oltre al contenuto del buffer (tipicamente testo), a
ciascuna finestra sono associate le seguenti
informazioni:

- **Modeline.**
- **Major Mode / Minor Modes** attive.
- Posizione del **point** nel buffer.



Informazioni per windows .

In generale è possibile avere più windows contenenti lo stesso buffer.



The screenshot shows the Emacs editor interface with two windows open, both displaying the same shell script buffer. The top window is titled 'emacs@aos-project' and the bottom window is titled 'U:**- example1.sh Bot L122 (Shell-script[sh])'. Both windows show the same code, which is a shell script for generating and checking URLs. The script includes functions like 'generate_urls_file()', 'generate_broken_urls_file()', 'check_for_broken_urls()', and 'recursive_iterate()'. The script is written in a mix of English and Italian comments.

```
File Edit Options Buffers Tools Sh-Script Help
PARTIAL_URLS_FILE="/tmp/partial_list_urls.txt"
BROKEN_URLS_FILE="/tmp/broken_urls.txt"

PUP_PATH="/home/leo/go/bin/pup"
XMLSTARLET_PATH=$(which "xml" 2> /dev/null)

# -----
main () {
    # -- requirements
    [ ! -f "$PUP_PATH" ] && [ ! -f "$XMLSTARLET_PATH" ] && echo "Missing requirements: xmlstarlet, pup" && exit

    generate_urls_file
    generate_broken_urls_file
}

# -----
# Extract url list
generate_urls_file() {
    rm -f $FULL_URLS_FILE
    rm -f $PARTIAL_URLS_FILE

    pushd $DATA_DIR > /dev/null
    # -- extract all urls
    recursive_iterate
    popd > /dev/null
    # -- force C locale to have byte-wise sorting
    LC_ALL=C sort -u $PARTIAL_URLS_FILE -o $PARTIAL_URLS_FILE
}

recursive_iterate() {
    for f in *; do
        # NOTE: maybe support more file extensions?
        if [ -f "$f" ] && [ "${f%.*}" = ".html" ]; then
            # -- file
            extract_all_urls $f $FULL_URLS_FILE $PARTIAL_URLS_FILE
        elif [ -d "$f" ]; then
            # -- dir
        fi
    done
}

# -----
# Iterate over all urls and check if they exist
generate_broken_urls_file() {
    rm -f $BROKEN_URLS_FILE
    check_for_broken_urls $FULL_URLS_FILE $PARTIAL_URLS_FILE $BROKEN_URLS_FILE
}

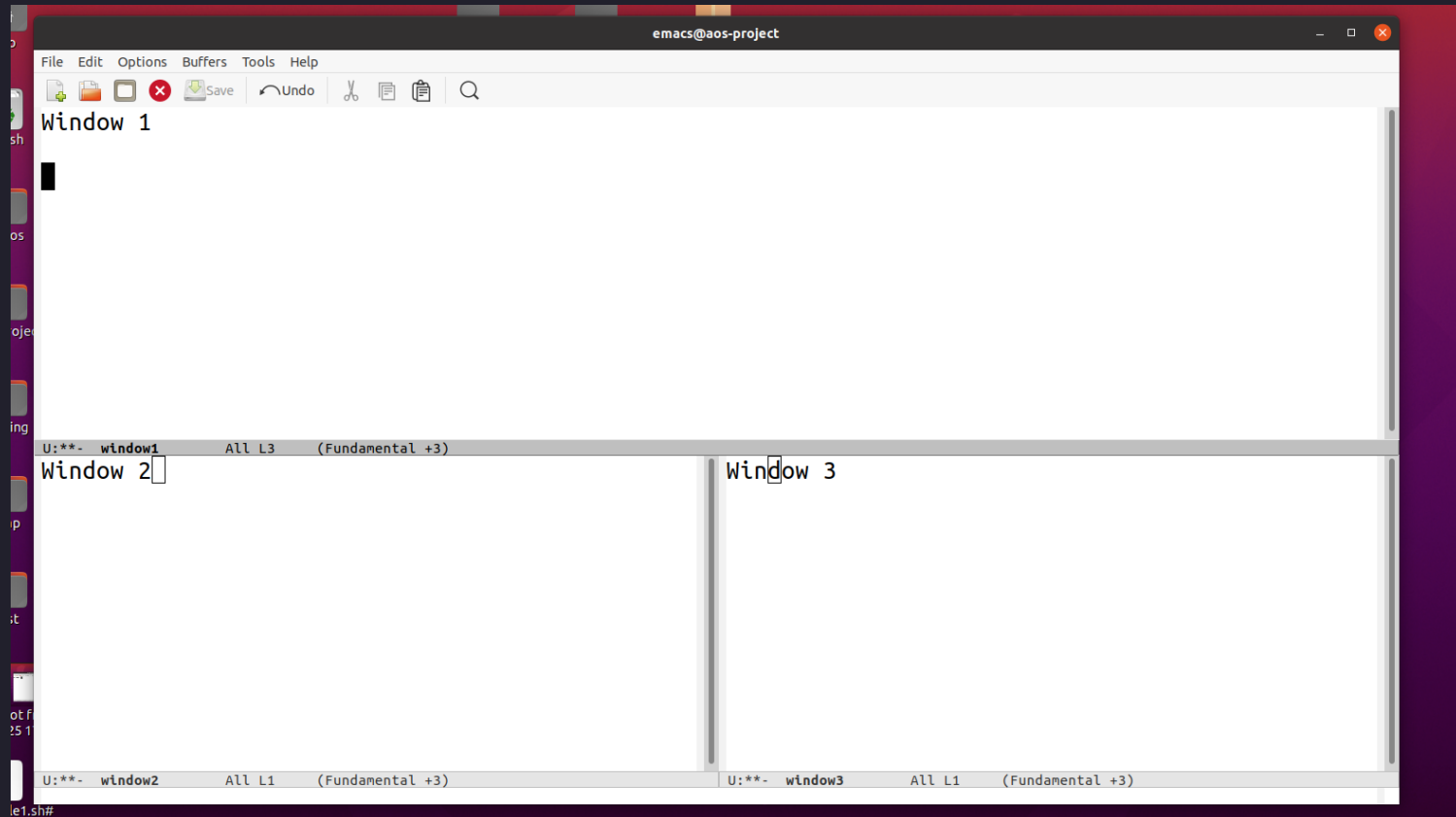
check_for_broken_urls() {
    local full_file=$1
    local partial_file=$2
    local err_file=$3

    while IFS= read -r url
    do
        # unescape HTML characters such as "&#39;" > "'"
        # requires xmlstarlet
        unescaped_url=$(XMLSTARLET_PATH unesc <<< "$url")
        if curl --output /dev/null --silent --head --fail "$unescaped_url"; then
            : # no-op
        else
            # -- url not found
            echo $url >> $err_file
            urlEscaped=$(sed 's/[^\w]/&#x[0-9a-f]/g; s/\^/\\^/g' <<< "$url") # escape it
            while IFS= read -r line
            do
                stringarray=( $line )
                echo " ${stringarray[1]}" >> $err_file
            done < <(grep -E "^$urlEscaped( | $)" "$full_file")
        fi
    done < $partial_file
}

main
```

Due windows aperte sullo stesso bufffer.

In ogni momento è presente un solo **cursore**, che rappresenta la **finestra attiva** (active window) su cui si sta lavorando.



Il cursore si trova nella finestra attiva (quella in alto).

Per gestire le finestre si utilizzano i seguenti keybinds

Keybind	Comando
C-x 2	split-window-vertically
C-x 3	split-window-horizontally
C-x o	other-window
C-x 0	delete-window
C-x 1	delete-other-window

Oppure i seguenti (meno utili)

Keybind	Comando
---------	---------

	delete-windows-on
--	-------------------

C-x ^	enlarge-window
-------	----------------

	shrink-window
--	---------------

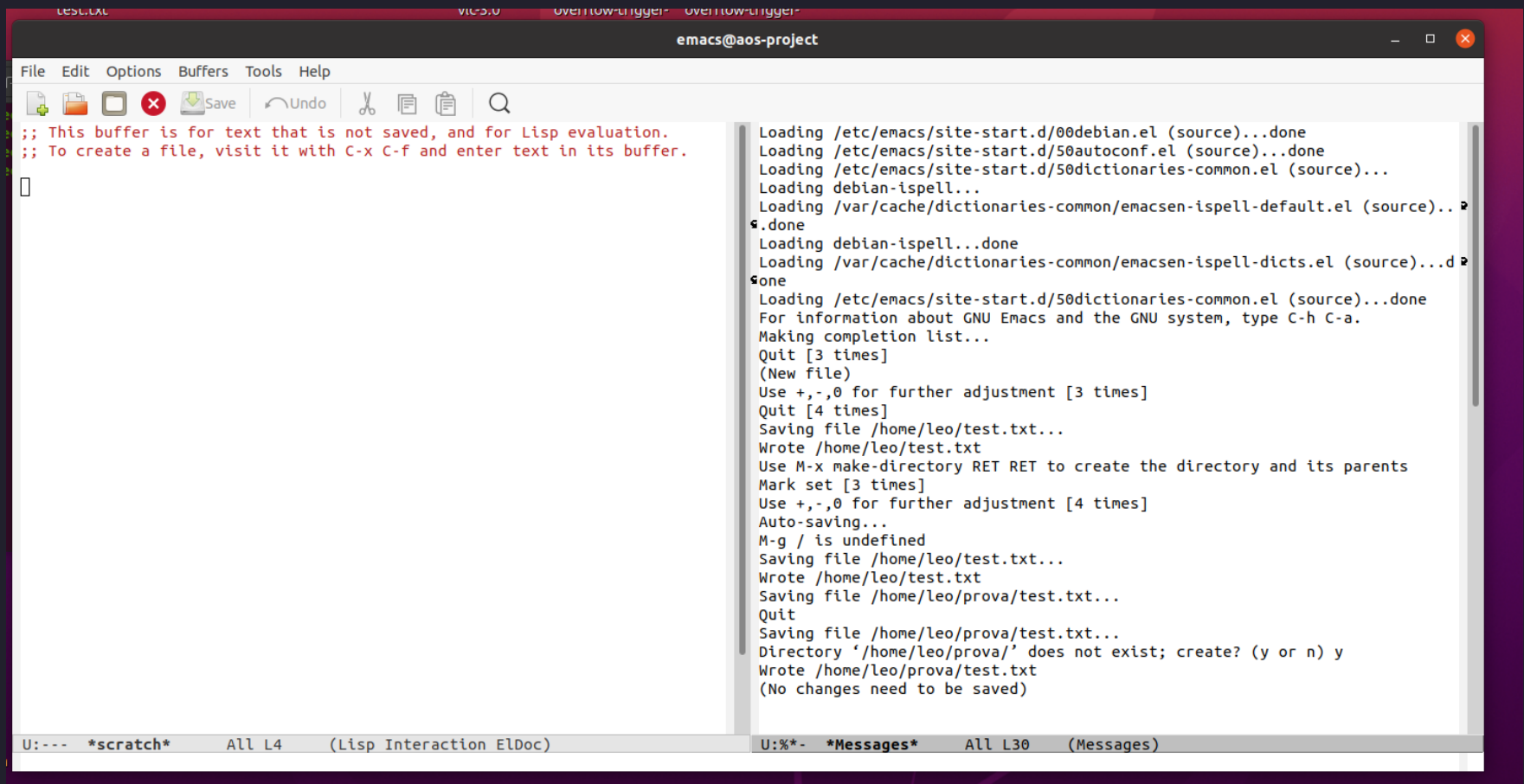
C-x +	balance-windows
-------	-----------------

BUFFERS

I **buffers** sono i contenitori di contenuto. In altre parole, sono i luoghi in cui lavoriamo sempre mentre utilizziamo Emacs.

Le finestre (e i frame) sono i modi per mostrare il contenuto di uno o più buffers.

Alcuni buffer sono "speciali", in quanto vengono creati e gestiti interamente da Emacs. Tra questi troviamo il buffer **Messages** e il buffer **scratch**.



The screenshot shows the Emacs editor window with the title bar "emacs@aos-project". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", and "Help". The toolbar contains icons for file operations and editing. The main window is split into two panes. The left pane shows the *scratch* buffer with the following text:

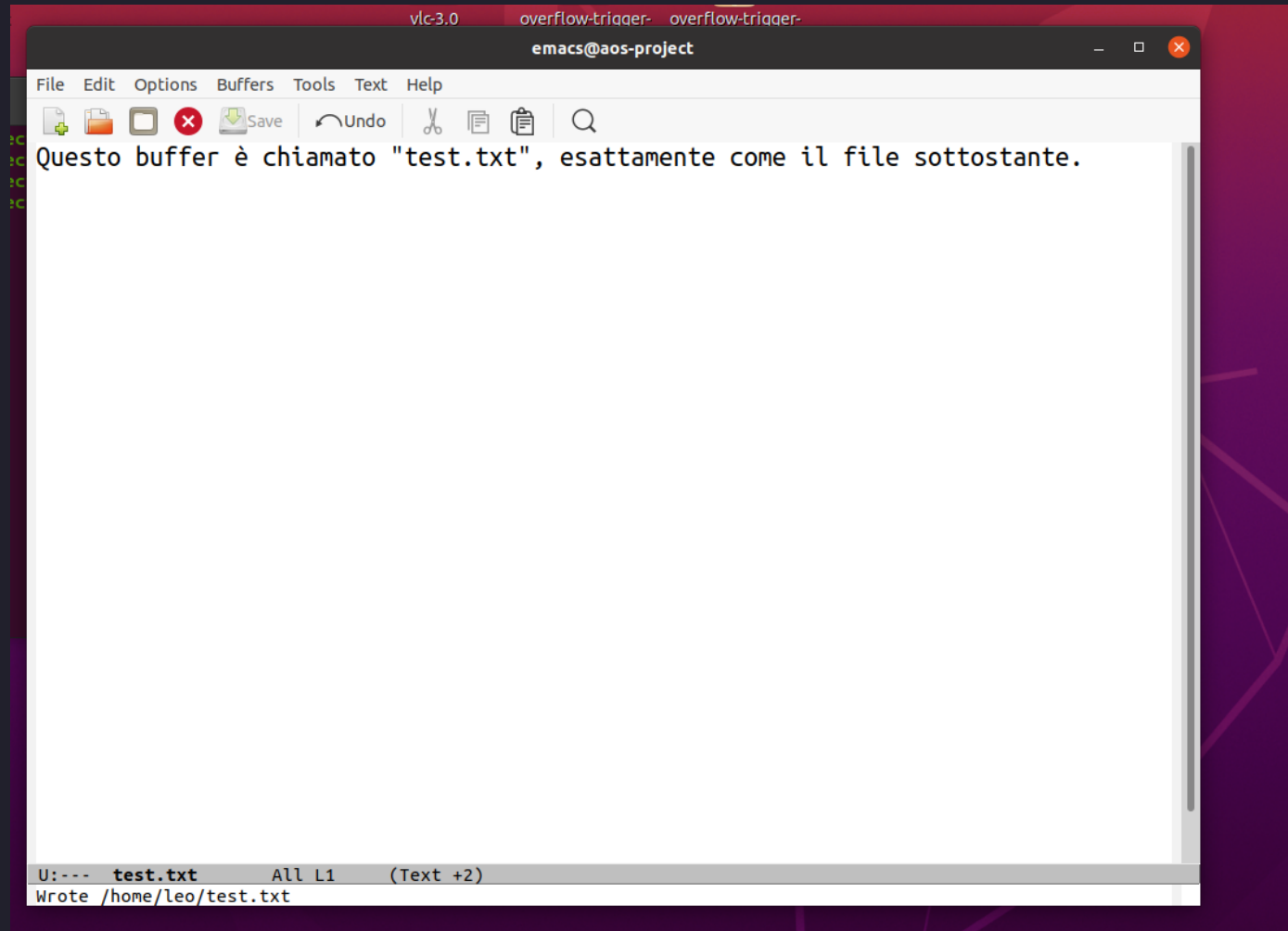
```
;; This buffer is for text that is not saved, and for Lisp evaluation.
;; To create a file, visit it with C-x C-f and enter text in its buffer.
[]
```

The right pane shows the *Messages* buffer with the following text:

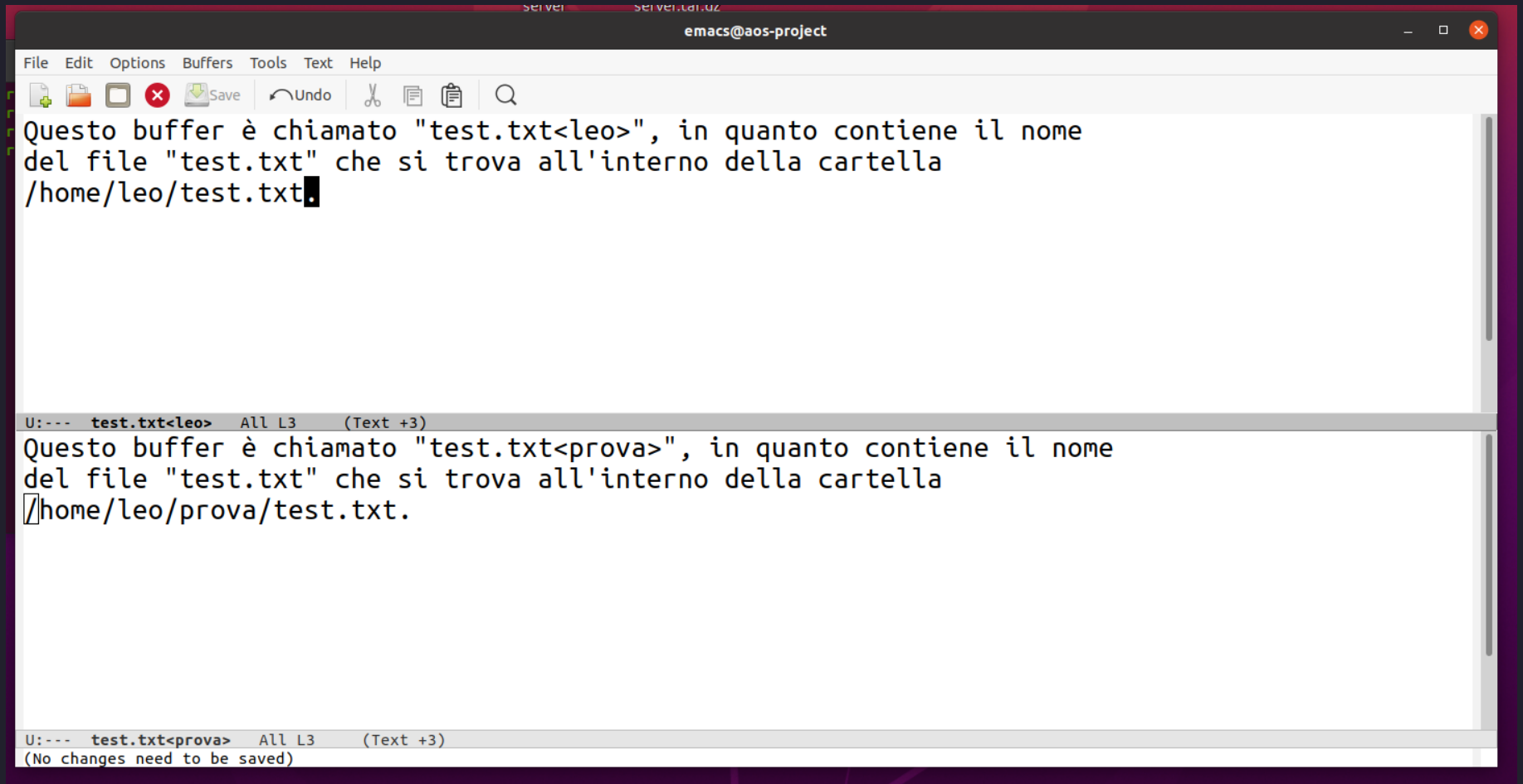
```
Loading /etc/emacs/site-start.d/00debian.el (source)...done
Loading /etc/emacs/site-start.d/50autoconf.el (source)...done
Loading /etc/emacs/site-start.d/50dictionaries-common.el (source)...
Loading debian-ispell...
Loading /var/cache/dictionaries-common/emacs-en-ispell-default.el (source)...done
Loading debian-ispell...done
Loading /var/cache/dictionaries-common/emacs-en-ispell-dicts.el (source)...done
Loading /etc/emacs/site-start.d/50dictionaries-common.el (source)...done
For information about GNU Emacs and the GNU system, type C-h C-a.
Making completion list...
Quit [3 times]
(New file)
Use +,-,0 for further adjustment [3 times]
Quit [4 times]
Saving file /home/leo/test.txt...
Wrote /home/leo/test.txt
Use M-x make-directory RET RET to create the directory and its parents
Mark set [3 times]
Use +,-,0 for further adjustment [4 times]
Auto-saving...
M-g / is undefined
Saving file /home/leo/test.txt...
Wrote /home/leo/test.txt
Saving file /home/leo/prova/test.txt...
Quit
Saving file /home/leo/prova/test.txt...
Directory '/home/leo/prova/' does not exist; create? (y or n) y
Wrote /home/leo/prova/test.txt
(No changes need to be saved)
```

The status bar at the bottom shows the current buffer is *scratch* and the *Messages* buffer is also visible.

Tramite il comando C-x C-f possiamo aprire il contenuto di un **file** all'interno di un nuovo **buffer**.



Il nome del nuovo buffer corrisponderà al nome del file, a meno di ononimi.



Due file con lo stesso nome, ma in diverse cartelle, aperti contemporaneamente.

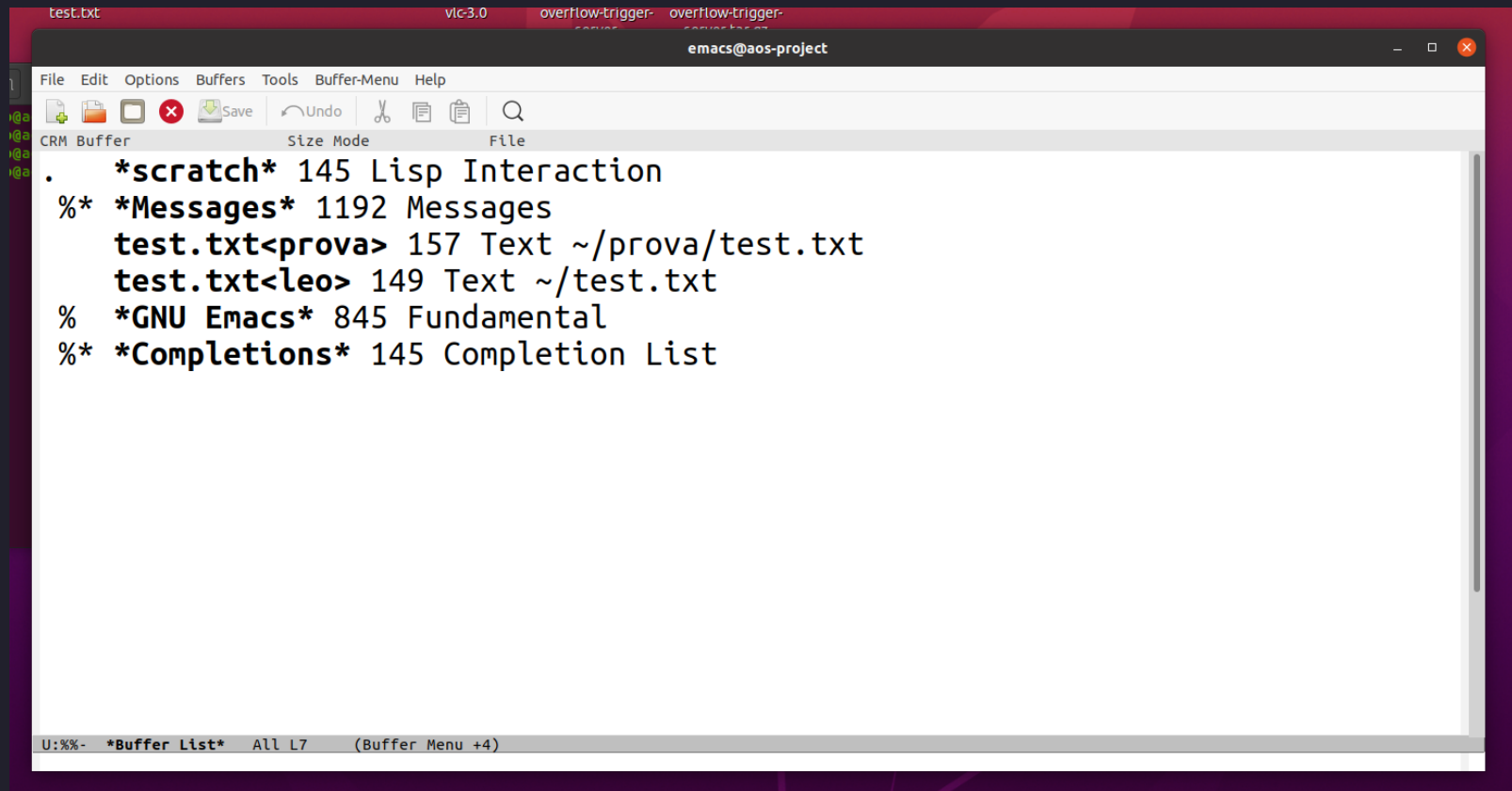
Tramite il keybind C - x b, che chiama il comando (`switch-to-buffer`), siamo in grado di

1. Cambiare il buffer attuale.
2. Creare un nuovo buffer senza associarlo ad alcun file.

In ogni dato momento vediamo solo una piccola parte dei buffer attivi. Per vedere tutti i buffer attivi ci sono tre modi diversi:

1. Tramite la **buffer list**.
2. Tramite il **buffers menu**.
3. Tramite il **buffer pop-up menu**.

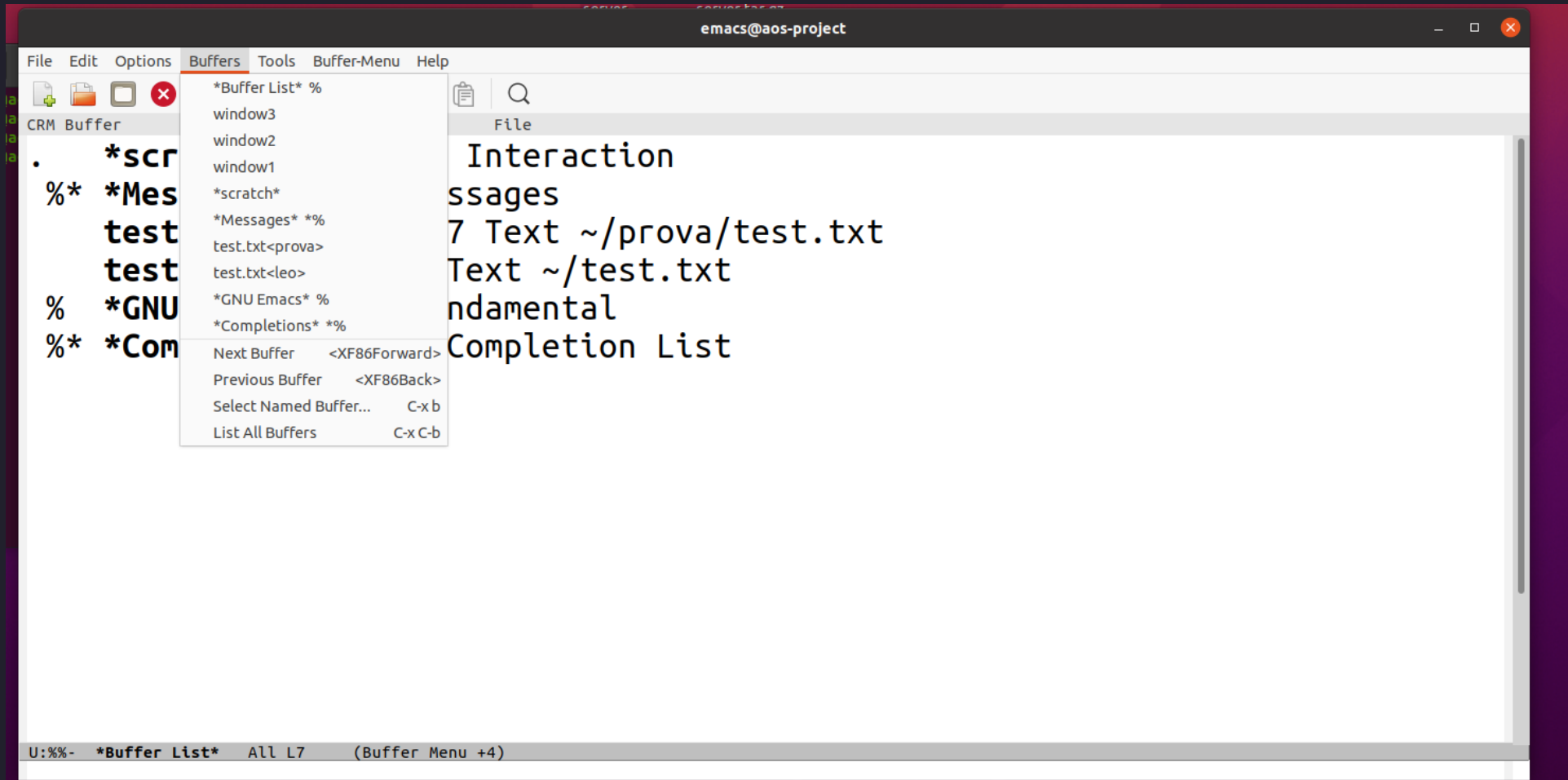
Per accedere alla **buffer list** possiamo utilizzare il keybind `C - x C - b`, o, equivalentemente, il comando `(list-buffers)`.

A screenshot of the Emacs editor's buffer list window. The window title is "emacs@aos-project". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Buffer-Menu", and "Help". The toolbar shows icons for "Save", "Undo", "Cut", "Copy", and "Find". The main content area displays a list of buffers with columns for "CRM", "Buffer", "Size", "Mode", and "File". The buffers listed are: ". *scratch*" (145 Lisp Interaction), "%* *Messages*" (1192 Messages), "test.txt<prova>" (157 Text ~/prova/test.txt), "test.txt<leo>" (149 Text ~/test.txt), "% *GNU Emacs*" (845 Fundamental), and "%* *Completions*" (145 Completion List). The status bar at the bottom shows "U:%%- *Buffer List* All L7 (Buffer Menu +4)".

```
test.txt vlc-3.0 overflow-trigger- overflow-trigger-
emacs@aos-project
File Edit Options Buffers Tools Buffer-Menu Help
CRM Buffer Size Mode File
. *scratch* 145 Lisp Interaction
%* *Messages* 1192 Messages
test.txt<prova> 157 Text ~/prova/test.txt
test.txt<leo> 149 Text ~/test.txt
% *GNU Emacs* 845 Fundamental
%* *Completions* 145 Completion List
U:%%- *Buffer List* All L7 (Buffer Menu +4)
```

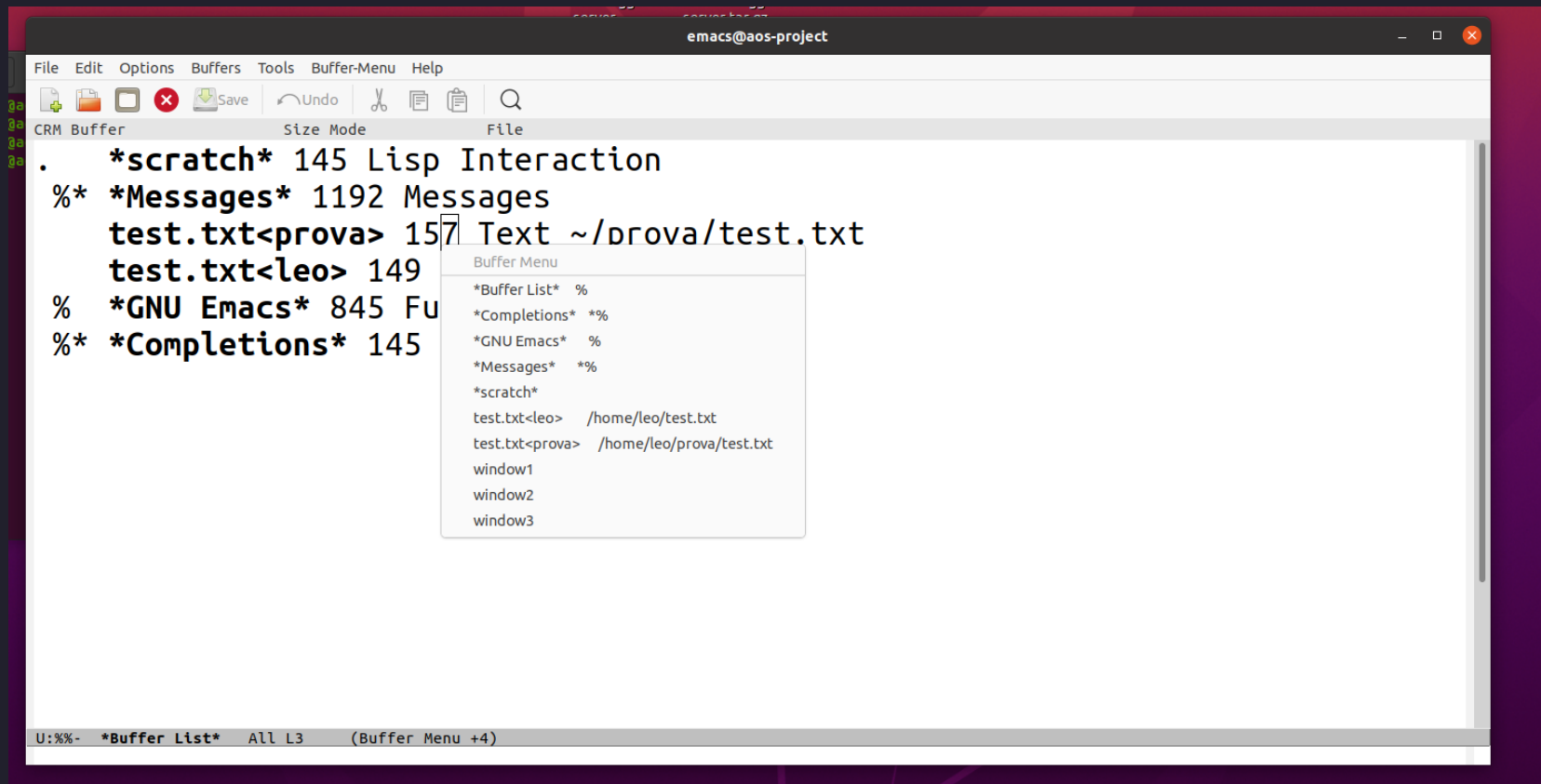
Buffer list.

Il buffers menu invece fa parte della GUI di Emacs.



Buffer menu.

Infine, il **buffer pop-up menu** fa sempre parte della GUI di Emacs, e può essere attivato premendo **CTRL + left mouse click**.



Buffer pop-up menu.

Una volta aperta la buffer list, possiamo eseguire vari comandi, tra cui

Keybind	Descrizione
1	metti in full screen il buffer puntato
f	mostra buffer puntato su finestra attuale
o	mostra buffer puntato su altra finestra
d, k	mark per deletion

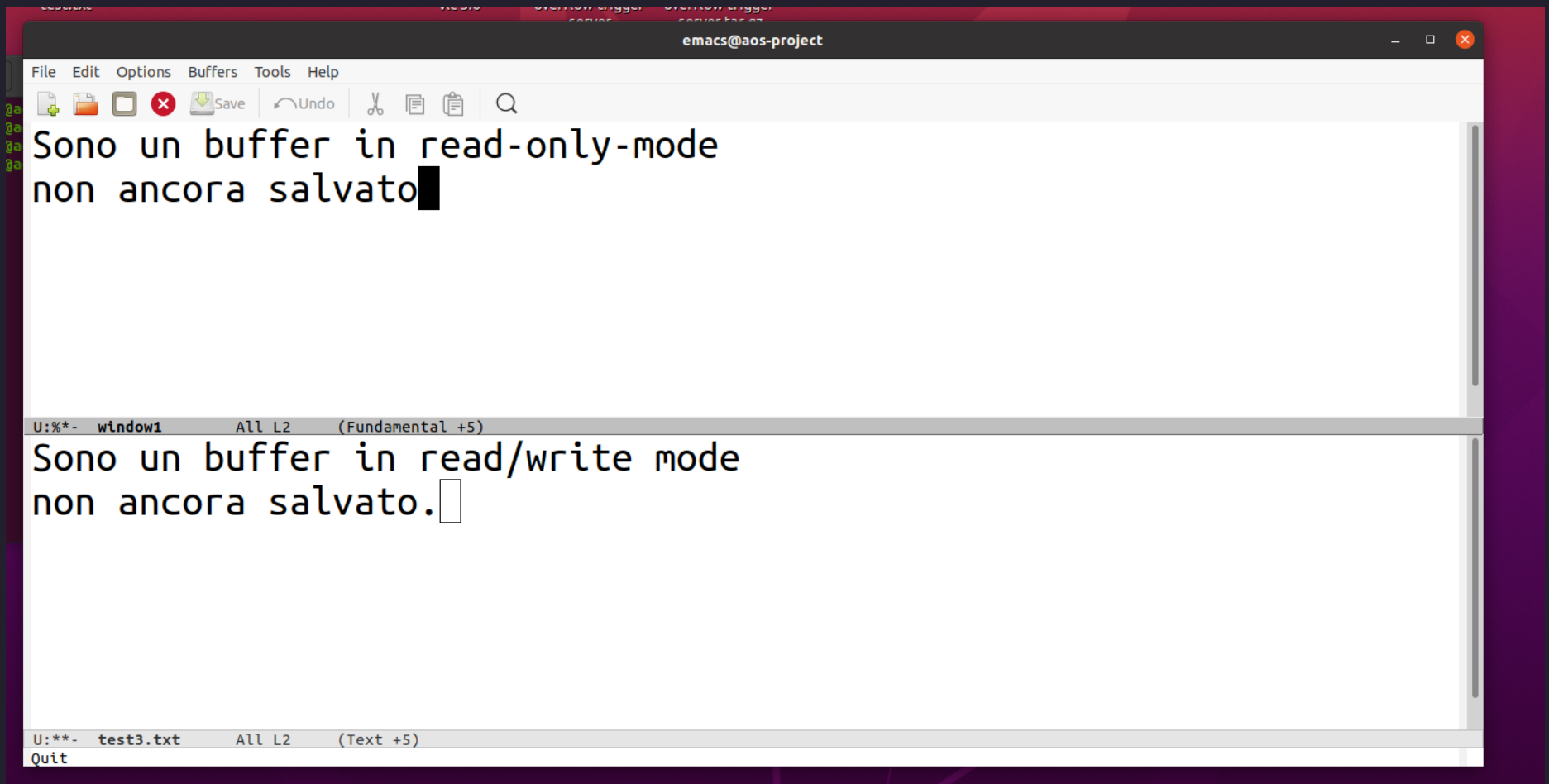
e anche

Keybind	Descrizione
s	salva il buffer
u	unmark
x	esegui comandi sui buffer marcati
q	quit

In generale, per gestire i buffers al di fuori della **buffer list** i seguenti keybinds sono utili

Keybind	Comando
C-x b	switch-to-buffer
C-x C-b	list-buffers
C-x k	kill-buffer
	rename-buffer
C-x C-q	read-only-mode

La **read-only-mode** è utile quando vogliamo essere sicuri di non modificare il contenuto di un file.



Read only mode

Altri keybind (meno utili) sono

Keybind	Comando
---------	---------

C-x s	save-some-buffers
-------	-------------------

	kill-some-buffers
--	-------------------
