GNU Readline Library

When using the GNU Readline Library, the user sees a prompt and types into what seems like stdout stream, but under the hood, Readline does a lot more than just reading characters.

Internally, Readline manages:

- A line buffer
    - A character array that holds the current input line
    - Cursor position is tracked as an index into this buffer

    ```
    char rl_line_buffer[MAX_LINE_LEN];
    int rl_point;  // current cursor offset
    ```

- A virtual cursor
    - This is not the actual terminal cursor
    - It’s Readline’s internal cursor, which moves as the user types, deletes, or navigates with arrows
    - Readline only updates the real terminal display when it has to, like after a key press or when “rl_redisplay()” is called
- Prompt state
    - Readline stores user’s prompt and the number of visible characters (especially for multiline prompts)
    - It keeps track of whether the user is mid-input or need to redraw the prompt

How This Is Used

Say the user is halfway through typing:

```
>>> ls -l /hom
```

Now, some background event prints “Download complete.” without telling Readline the user is on a new line and to redraw:

- That message would appear in the middle of the prompt
- The terminal cursor and Readline’s internal buffer would become desynchronized

In order to avoid desynchronization, do:

```
printf("\n[Download complete]\n");
rl_on_new_line();  # acknowledge output break
rl_redisplay();  # redraw line cleanly under printed message
```

Terminal Cursor VS Virtual Cursor

Two elements, Terminal Cursor and Virtual Cursor, should be understood to learn how cursors work:

- Terminal Cursor (visual)
    - Managed by the terminal emulator
    - Usually represented as a blinking character on the screen
    - Affected by direct outputs like “printf()”, “write()”...
- Virtual Cursor (logical)
    - Managed by Readline library
    - Tracks position in the “rl_line_buffer”
    - Doesn’t directly affect terminal output until “rl_redisplay()” is called

Most of the time these two match, but if an asynchronous operation modifies the Terminal Cursor, a synchronization is needed in order to display the terminal output correctly.