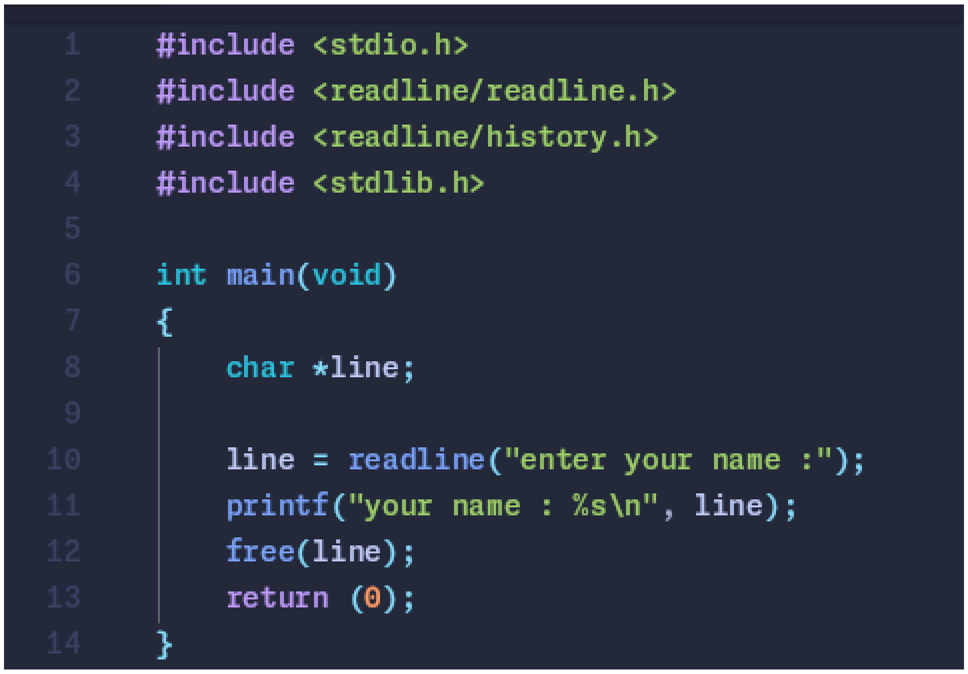
readline() :

prints a prompt and then reads and returns a single line of text from the user. The line readline returns is allocated with malloc (); you should free () the line when you are done with it.

char *line = readline ("Enter a line: ");



```c
#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <stdlib.h>

int main(void)
{
    char *line;

    line = readline("enter your name :");
    printf("your name : %s\n", line);
    free(line);
    return (0);
}
```

cc -Wall -Wextra -Werror main.c -lreadline

rl_clear_history() :

rl_on_new_line() :

Tell the update routines that we have moved onto a new (empty) line, usually after ouputting a newline.

rl_replace_line() :

int rl_replace_line(const char *text, int clear_undo);

The `rl_replace_line` function replaces the current line buffer with the string provided in the `text` parameter. If `clear_undo` is non-zero, the undo list is cleared, meaning that the replaced line cannot be undone.

evaluation :

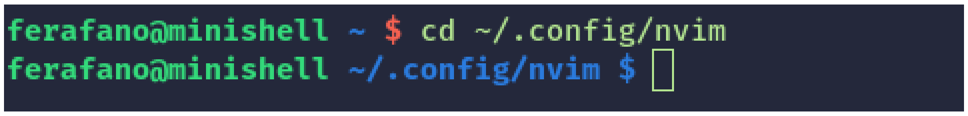
- after a command give back the prompt
- if the command doesn't exist it must return a proper error
- support the original command path
- support multiple flag
    1. like ls -la
    2. like ls -l -a
    3. like ls -l   -a
    4. like   /bin/ls   -l   -a
- exit : quit the shell
- echo must supports " " ou not , erro if one "
- cd - : return to last directory
- env : display as key=value
- setenv FOO bar or setenv FOO=bar : create a new key=value in env
- echo $FOO : display the value of the key FOO
- /usr/bin/env". Minishell must send the appropriate environment to ran binaries. /usr/bin/env must display environment including FOO and its value bar
- unsetenv FOO : remove the key=value in env … if not in env , do nothing
- if unsetenv PATH the command shouldn't work , but if we "$> setenv PATH "/bin:/usr/bin" or "$> setenv "PATH=/bin:/usr/bin" , the command should rework
- emacs must run /usr/bin/emacs … if unsetenv PATH , it shouldn't work
- but even if unsetenv PATH , /bin/ls should work
- if nothing , do nothing , give back the prompt
- single space , do nothing , the command must give back the prompt
- space and tabulation , do nothing and give back the prompt

bonus :

- ctrl+c give stop the current cmd process then give back the prompt … if not cmd running or just prompt , just give back the prompt …
- - Create a new folder /tmp/bin/ and add this folder to the PATH environment variable. Create a program named 'test_exec_rights' inside that folder that will just display 'KO'. Give this program the following rights 644 (meaning no execution

rights). From another folder, run the following command "$> test_exec_rights". Check that the minishell refuses to run the program because of the missing execution rights.

- Type the following beginning of command "$> ec", then press tabulation. The minishell must complete the command into "$> echo"
- - Run the following command "$> echo TOP ; ls ; echo MIDDLE ; ls ; echo BOTTOM". The 5 commands must be executed without any errors in the order they were written. - Run the following command "$> ;". The minishell must either do nothing and give the prompt back or display a syntax error and give the prompt back.
- If the project has other operational bonuses, you can evaluate them and grade them in this section : ex color etc …



```
ferafano@minishell ~ $ cd ~/.config/nvim
ferafano@minishell ~/.config/nvim $ 
```