MINISHELL : LEXER → EXPANDER → PARSER → EXECUTOR

main()

init_shell()

- (1) print_welcome()
- (2) init_env()
- (3) init_builtins()
- (4) init_signals()
- (5) get_path()

shell_loop()

- prompt = readline()
- handle_eof()
- check_prompt()
  - add_history (prompt)
  - ft_strtrim()
    → Elimino espacios iniciales y finales, si solo hay " ", NULL
  - check_quotes()
    Control comillas emparejadas y pares.
    Con flags, controlando qué es Texto y qué es comillas.
    Cada vez que encuentro una comilla,
    - s, comilla simple (39, '\'')
    - d, comilla doble (34, '\"')

    Desde que abro una comilla, hasta cierre es Texto.
- lexer()
  - Tokenize_prompt()
    - next_node_end()
    - ft_strndup()
    - append_node()
  - analyze_Tokens_type()
    - is_pipe : "|"
    - is_redirect_in : "<"
    - is_redirect_out : ">", ">|"
    - is_append : ">>"
    - is_heredoc : "<<"
    - is_command
    - is_syntax_error
- expander()
  - expand_variables()
    - handle_single_quotes()
    - handle_dbl_quotes()
    - handle_reg_chars()
  - expand_status()
    - ft_itoa(status)
    - expand_status_str()
  - word_splitter()
    - count_parts()
    - process_words()
  - quote_cleaner()
    - clean_line_quotes()
- parser()
- execute()

free_shell()

- free_matrix(shell->env)
- free_matrix(shell->path)

NOTAS HEREDOC

1) Controlar señales dentro del heredoc
   - SIGINT (Ctrl + C)
     cat << EOF
   - SIGQUIT (CTRL + \), no controla Core Dump.

2) Crear archivos temporales en "/tmp/.Temp",
   un archivo para cada heredoc con nombre ÚNICO.

3) Capturar entrada en el heredoc con "readline" hasta
   que entra el "delimiter".
   - Si el delimiter va entre comillas, no expando.

4) Controlar expansión de variables $VAR, sin control de comillas.

5) Escribir cada línea en el heredoc.

6) Eliminar archivo.

(*) Puede haber varios heredocs, cat << EOF1 | cat << EOF2