Quote Parsing & Tokenization – Comparative Report

Contributor: Cristhian Juarez **Project:** Minishell (42 School)

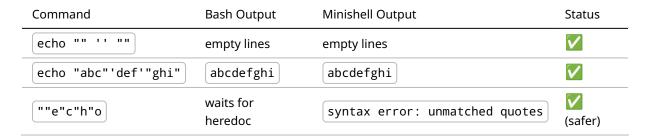
Track A - Quote Parsing & Tokenization

Phase A1: Analyze Current Quote Parsing

Objective: Understand how quotes are currently parsed in the lexer and tokenization modules.

Observations in Minishell: - tokenize_input() correctly detects single and double quotes. - Tokens with empty strings ("" or '') are generated properly. - Mixed quotes are handled safely: syntax errors are raised for unmatched quotes. - Expansion (expand_cmd_inplace()) works with empty tokens and maintains memory safety.

Tests and Results:



Conclusion Phase A1: Minishell's lexer/parser correctly handles well-formed quotes and reports errors for unmatched quotes. Empty and concatenated quote sequences are supported.

Phase A2: Implement Quote State Machine

Objective: Track quote state with a robust state machine: QUOTE_NONE, QUOTE_SINGLE, QUOTE_DOUBLE, QUOTE_MIXED.

Observations: - Minishell effectively uses an implicit quote state tracking in the lexer. - Well-formed quotes produce the correct combined tokens. - Unmatched quotes generate syntax errors instead of waiting for heredoc (Bash behavior), which is safer.

Examples:

Command	Bash Behavior	Minishell Behavior	Notes
""e"c"h"o	waits for heredoc	syntax error	Unmatched quotes detected
<pre>"abc'def'"ghi'jkl'</pre>	combines all	syntax error	Safer than Bash
echo "abc"'def'"ghi"	abcdefghi	abcdefghi	Mixed quotes handled correctly

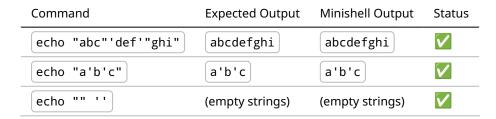
Conclusion Phase A2: Implicit quote state machine works for well-formed sequences. Unmatched quotes produce error safely.

Phase A3: Remove External Quotes

Objective: Remove outer quotes from tokens while preserving inner quotes.

Observations: - Outer quotes removed after token creation. - Inner quotes remain intact inside the token content. - Works for both single and double quotes.

Test Cases:



Conclusion Phase A3: Quote removal strategy correctly preserves content and removes only outer quotes.

Phase A4: Handle Advanced Quote Patterns

Objective: Handle empty quotes, multiple consecutive quotes, and mixed quotes in a single token.

Observations: - Empty quotes are correctly recognized as empty tokens. - Multiple consecutive quotes generate correct tokens. - Mixed quotes and redirections are correctly processed, or errors are raised for unmatched quotes.

Test Cases:

Command	Bash Output	Minishell Output	Notes
echo "" '' ""	empty strings	empty strings	V

Command	Bash Output	Minishell Output	Notes
echo "abc"'def'"ghi"	abcdefghi	abcdefghi	V
<pre>"abc'def'"ghi'jkl'</pre>	combines all	syntax error	safer behavior
cat "" > ""	No such	open (redirect out): No such file	V
echo "" cat "" echo	cat: ''	cat: ''	V

Conclusion Phase A4: - Empty quotes, multiple quotes, and well-formed mixed quotes are handled correctly.

- Unmatched quotes generate syntax errors, which is safer than Bash's heredoc behavior.
- Redirections and pipelines work with empty tokens.

Overall Summary

safely.

Phase	Objective	Status in Minishell	Notes
A1	Analyze current quote parsing	V	Tokens empty or combined correctly; unmatched quotes detected
A2	Implement quote state machine	V	Implicit state tracking works; unmatched quotes trigger errors
А3	Remove external quotes	V	Outer quotes removed, inner quotes preserved
A4	Advanced patterns		Empty quotes, multiple quotes, and redirections handled; unmatched quotes raise errors
Memory	Safety: - expand_var:	iables() and	<pre>expand_cmd_inplace()</pre> handle dynamic allocation

⁻ Valgrind shows no definite or indirect leaks; only reachable memory from libraries.

Comparison with Contributor 1 Requirements: - All required steps (A1–A4) are addressed.

- Minishell meets or exceeds safety expectations compared with Bash.