Ariel Ahdoot and Timothy Goetjen

Systems Programming Assignment #1

Tokenizer.c Documentation

Tokenizer.c takes in command line arguments and returns the type of each token followed by that token in quotations on a newline.

The program utilizes many functions in order to determine type of token and return the type and the token value to console output.

Character arrays and arrays of character arrays are utilized to facilitate the steps within each function.

Types of tokens:

Decimal

Hexadecimal

Octal

Floating point

Word

Keyword

Operators (various)

Strings

Comments

Escape Characters

Malformed objects (invalid input)

Whitespace

Whitespace and comments are not considered tokens and are ignored when traversing the input field of the tokenizer and likewise have no output to the console.

Escape characters are represented by their hexadecimal values in brackets.

Our code is prepared to deal with escape characters when the literal character i.e. an actual backspace escape character is input to the command line from an input file that was created from

a C file that contained a printf statement that printed out the escape character. While our code is prepared to deal with escape characters, the input of them as a '\' followed by the corresponding letters or numbers is considered malformed followed by either a word or the appropriate type of number.

Example "\n" outputs:

Malformed object '\'

Word "n"

The following are defined as operators: ()[].*&-!~/%+<>&^|,=?:

Anything outside of those is considered malformed, except for double and triple character operators, which are considered operators and dealt with accordingly.

Any number of consecutive or separated operators will be broken up into the appropriate single, double, or triple operator tokens, with the exception of '/*', which is taken as the start to a multi-line comment and is considered malformed if there is no corresponding end to the comment, '*/'.

A hexadecimal number must start with '0x' or '0X' and contains valid hex alphanumeric characters immediately following that, otherwise the '0x' or '0X' is returned as a malformed object, and the remainder of the alphanumeric characters are dealt with according to word and number types.

Any of the following inputs regarding floating points are considered malformed:

"1e"

"2E"

"3e-"

"4e+"

"2E+"

"2E-"

"1."