A REAL TOKENIZER

A Tokenizer is a program that breaks the input text string into smaller tokens. Each token would have a specific meaning. Their meanings would be interpreted by the tokenizer and returned with the tokens.

Files Included in this project:
+ hashtable.h : the implementation of the hash table to be later used by the C-operator dictionary and C-keyword dictionary
+ tokenizer.c : the main program of the tokenizer

How To Compile the Tokenizer:
In the project directory, execute the command:
gcc ./tokenizer.c -o ./tokenizer -Wall

How To Use the Tokenizer:
./tokenizer “the_string_that_is_tokenized”

What types of tokens the tokenizer recognizes:
+ decimal number
+ octal number
+ hexadecimal number
+ floating point number
+ malformed type
+ word and C-keyword
+ operator and C-operator
+ escape sequence (return an error)

What special features the tokenizer has:
+ All C style comments would be skip
“// comment” skip all characters until the tokenizer encounters a new line character
“/* comment */“ skip all characters between the comment signs “ /* ” and “ */ ”
+ All word that is a C-keyword will be recognized separately.

The implementation of the tokenizer:
The tokenizer acts like a parser, it has a copied string from the argument and a character pointer points to the current position of the string in the tokenizing process. Every time the program interprets the next string as a new token that has a specific meaning it would return the token, the pointer shifts to the beginning character of the following token to be processed.

There are total 7 meaningful types of token as mentioned above. The most special case is the operator tokens, because we want to know what the exact meaning of the operators is. Instead of enumerating all 42 cases of operator, we add a C-operator-dictionary implemented as a Hash Table using separate chaining technique. The same thing had been done with the C-keyword recognition feature. A hash table could make the searching process becomes faster and in some cases, the complexity is O(1).

Examples:

Input:
./tokenizer "abc 123 1.23 1.23e9 1.23e-4 0.123 0123 01238 0x1a2b3c //abc
0x /*cba*/ .123” “/ *= if”

Output:
word "abc"
integer "123"
float "1.23"
float "1.23e9"
float "1.23e-4"
float "0.123"
octal "0123"
integer "01238"
hex "0x1a2b3c"
mal "0x"
float ".123"
solidus "/"
asterisk equals “*=“
C-keyword “if”