Arifur Rahman

Systems Progamming PA1

My Tokenizer program runs as mentioned below

Input argument is a string on the command line. My program then performs a copy on the string and has it stored into a data structure called a Token. Then my program analyzes where each char belongs and designates a specific location for it (parsing). Each token is parsed out with the TKGetNextToken function as strings containing the Token_Type and the Token_Content.

The output will look something like this:

Hex "0x42"

Word "Bob"

Int "42"

Token strings will be destroyed whenever the next token is being parsed. After all tokens have been parsed, the Tokenizer object will be destroyed using the TKDestroy function.

Parsing:

The Tokenizer struct stores the complete command line string, and an int. The int will keep track of the programs current position through the entire string. String will remain until the entire Tokenizer object is destroyed.

TKGetNextToken will work as a recursive function which uses the current position in the Tokenizer as a starting point for parsing the next token. The general parsing algorithm is as follows:

if current character ==[Specifier]

Loop until the character isn't valid for its specifier requirements, while counting token length

Create a new string large enough to hold the Token_Type and Token_Content (size known from length calculated by looping, Token_Type size is known and added)

Copy the Token_Type into the created string

Concatenate the Token_Content and end of string character

else check if the current character matches another class

The Specifiers available are

Words

Sub Set: C Key Words

auto

break

case

char

const

continue

default

do

double

else

enum

extern

float

for

goto

if

int

long

register

return

short

signed

sizeof

static

struct

switch

typedef

union

unsigned

void

volatile

while

Numbers

Positive Ints

Floats are accepted in the following forms:

1.1

1.00e-12

1.0

1.e10

Floats of the form 1.3f are read as

Float "1.3"

Word "f"

Oct are accepted in the form "0###" of any length

0 is accepted as an Oct

089 is accepted as an Oct: This style of invalid input is up to the programmer to avoid.

Hex are accepted in the form of 0x with upper and lower case letters with characters of 0-f

C operators

Any of the following qualify as a C operator:

+, ++, +=

-, --, -=, ->

*, *=

/, /=

&, &=, &&

|, ||, |=

%, %=

!, !=

~

:

?

<, <<, <=, <<=

>, >>, >=, >>=

=, ==

^, ^=

,

[ ]

( )

{ }

White space characters are

0x20 space

0x09 tab

0x0a newline

0x0d carriageReturn

These types of characters will cause the program to recurse into the function again, until it finds the next available token.

If a character does not match any specifier, the output would be:

Unknown Input [0x##]