Rob Williams and Sangini Shah
pa1 cs214

The Tokenizer runs as follows:
A string is input as a command line argument. This string is copied and stored into a data structure called a Token and then is parsed on a token by token basis until the whole command line string has been read. Tokens are parsed out with the TKGetNextToken function as strings containing the Token_Type and the Token_Content, which produces output such as:

Hex "0x42"
Word "Bob"
Int "42"

these token strings are then destroyed as the next one is parsed. Once all tokens have been parsed, the Tokenizer object is destroyed using the TKDestroy function.

Parsing:
The Tokenizer object (I use object to refer to the struct, not an OOP object) stores the complete command line string, and an int that keeps track of the current position through the string. This string remains until the entire Tokenizer object is destroyed and is not mutated during the course of the program's implementation.

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

if the current character is [Specifier]
Loop until the character invalidates its specifier requirements, count token length
Create a new string large enough to hold the Token_Type and Token_Content (size know 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
and these characters cause the program to recurse into the function again, in search of the next available token.

In the case where a character doesn't match a specifier, it is written in the form of
Unknown Input [0x##]