C-net Programming Final Report

Rediet Bekele - rsb2179	- Manager
Kidus Mulu - km3533	- Language Guru
Kingsley Neequaye - kn2427	- Tester
William Oseghare -who2103	- Systems Architect
Bruk Zewdie - bbz2103	- Systems Architect

1. Introduction

C-net is a language for network programming mostly based on the C programming language. It was developed to create a seamless way for users to implement network/file programming through succinct code and easy interaction with files and sockets. C-net provides an abstract wrapper for files and network sockets as objects for reading and writing along with built-in methods for performing common manipulations. In doing so, the language simplifies I/O for succinct and clear code. Furthermore, C-net discards the complex semantics of dynamic memory allocation in C by abstracting pointers for the user. As a result, most data on C-net is stored on the heap, except for primitives and the references to the objects, similar to the Java programming language. In addition, C-net provides a highly intuitive and convenient interface to strings that is used for all I/O operations and automatically allocated/freed without explicit invocations from the programmer. For non-primitive objects except strings, however, C-net does not have automatic garbage collection and therefore the user is required to free all allocated objects, although the semantics for allocating and freeing objects is quite straightforward.

C-net Tutorial

1.1. Environment Setup

The first step in interacting with C-net is to download the code and tests for the compiler from GitHub at the following link: https://github.com/Bruk3/C-net/tree/main. You will also need to have LLVM, OCaml and OPAM installed on your local environment.

1.2. Compilation Guide

First simply run

```
make
```

to generate the cnet.native compiler. Once the compiler is built, all of the unit tests and the integration tests will be executed.

Take a look inside **runtests.sh** for the variables which need to be set to the correct system paths of various programs, and set them accordingly. The following command runs all tests.

```
./runtests.sh
```

When you run the command above, all the tests that came with the compiler will be run and should pass. If they do not all pass, go back to section 2.1 Environment Setup and ensure that your environment is set up correctly.

To compile and create C-net source code after making sure that your environment is set up, follow these steps:

- Make sure that the cnet.native compiler is created. If it is not, follow the instruction above to create the compiler using the make command.

- Write source code by creating a file with a .cnet extension. Make sure that your file is in accordance with the rules outlined in the LRM below.
- You can compile and run an executable file from your source code as follows

```
./ccnet <file name>.cnet

./<file name>
```

Note: The compiler assumes that the llvm compiler and interpreter are part of the system's path.

1.3. Language Tutorial

GCD algorithm

The following is an example of how you can create a cnet program. First create a file and name it `gcd.cnet` Your file can contain the following code

```
int gcd(int a,int b){
       while(a!=b) {
       if(a>b)
              a -= b;
           else
                b -= a;
       }
       return a;
}

int main (){

       int b = gcd(12,6);

       stdout.writeln(soi(b));

       // soi refers to the cnet function that converts integer to string

       return 0;

}
```

Compile and run the given code using the following commands:

```
$ ./ccnet gcd.cnet

$ ./gcd

$ 6
```

netcat--

Now for a more network oriented program, we can write a simple chat server that acts like a very stripped down version of netcat.

The server:

```
int main (string[] argv){
	socket lsock = nopen("", argv[1], "tcp", "listen");
	socket clntsock = lsock.naccept();
	string received_str = "temp";
	while(received_str.slength() > 1) {
		clntsock.writeln(stdin.readln());
		received_str = clntsock.readln();
		stdout.writeln(received_str);
	}
	delete clntsock;
	delete lsock;
	return 0;
}
```

The client can be written in much the same way, simply using "connect" instead of "listen" as the socket type. For a connection socket, there is no need to accept a connection and the socket returned from the nopen call will be a connected socket. After compiling the two sources, you can connect them to each other and see that it does indeed produce the desired result.

```
$ ./ccnet -o cnetcat-serv cnetcat-serv.cnet
$ ./cnetcat-serv 10000
Hello from the server
Hello from the client
```

```
$ ./ccnet -o cnetcat-cli cnetcat-cli.cnet
$ ./cnetcat-cli 10000
Hello from the client
Hello from the socket
```

Here the client sends the message “Hello from the client” and receives the message from the server saying “Hello from the server”, and vice versa for the server. As soon as the server receives a new line only (no message), it cleans up its sockets and exits.

cat without concatenation

The final tutorial program, an introduction to File IO in C-net, will simply echo the contents of a file into the terminal (or stdout to be more precise). The source code is shown below:

```
int main (string[] argv){
	string filename = argv[1];
	file f = fopen(filename, “r”);
	string s = f.readln();
	while(s.slength() > 0)
	{
		stdout.writeln(s);
		s = f.readln();
	}
	return 0;
}
```

This program reads the target file specified by the first command line argument and prints it out to standard out. Although this example is good for illustrative purposes, C-net provides a simpler call for use cases like the one above (*readall*) that reads from a file/socket until the end of file is reached, and could simplify our already simple cat program even more. The source code of using the readall function would be as follows:

```
int main (string[] argv){
        string filename = argv[1];
        file f = fopen(filename, "r");
        stdout.writeln(f.readall());
        return 0;
}
```

The program can then be run at the command line as follows:

```
$ ./ccnet cnetcat.cnet # no -o flag outputs to the filename without .cnet
$ ./cnetcat cnetcat.cnet
 int main (string[] argv){
        string filename = argv[1];
        file f = fopen(filename, "r");
        stdout.writeln(f.readall());
        return 0;
 }
$
```

That concludes our small tutorial. More C-net programs can be found in our test files and demo http server.

2. C-net Language Reference Manual

1. Lexical Conventions

1.1 Tokens

There are five classes of tokens: identifiers, keywords, literals, operators and delimiters. White space (tabs, new lines, and blank spaces) and comments are ignored, except when used to separate tokens.

1.1.1 Identifiers

Identifiers are names that are associated with a certain value (in the case of a variable) or an operation (in the case of a function). They are any sequence of letters, digits, and '_' that **do not** begin with a digit.

Examples of valid identifiers in C-net include

```
myvar, my1var, _myvar, name2, …
```

while invalid identifiers are any names that start with a digit or contain other characters such as

```
5_myvar, my-var, ...
```

1.1.2 Keywords

C-net has a list of identifiers that are reserved for use by the language and cannot be used otherwise. Below is a list of keywords:

```
int void new char if delete float else return struct

for string while socket break file continue
```

1.1.3 Literals

Literals are values embedded into the program. There are two types of literals: numeric literals (integer, character, and float) and string literals.

Numeric literal

Integer literals

An integer literal is a sequence of digits in decimal that may optionally be preceded by '-' to signify a negative value. The matching regex is integer = ['0'-'9']+

Character literals

A character literal is a single printable ASCII character (values 32-126 inclusive) enclosed in single quotation marks (with a few exceptions). For computational purposes such as addition, subtraction or comparison with other characters, it can be treated as the number entry of the character in the ASCII table.

Non-printable characters can be represented with a backslash character '\' followed by three digits signifying the octal value of the ASCII table entry number for the desired character. Some examples are presented below.

```
'\012'   // ASCII new line (line feed) character

'\009'  // ASCII horizontal tab character
```

N.B. backslash ('\') denotes an escape sequence for non-printable characters.

C-net provides shorthand notations to represent some special characters. Below are the acceptable shorthand char literal representations in source code translated to their corresponding octal representation in the right column:

```
// Escape sequences          //Translated character representation

'\n'                         '\012'

'\t'                         '\009'

'\\'                         '\134'

'\0'                         '\000'
```

Floating point literals

A floating point literal is a sequence of digits (with an optional preceding '-') followed by a single decimal point, and possibly followed by a sequence of digits. Regular expression is (['0'-'9'])+'.'(['0'-'9'])* (exp)?) | ((['0'-'9'])* '.'(['0'-'9'])+(exp)?) | ((['0'-'9'])+exp)

For example,

```
3.1415 and 20. //valid floating point literals

.15 and 1.16.2021 //not valid floating point literals
```

String Literals

A string is a sequence of character literals. For the sake of tokenization, they are a sequence of ASCII printable characters (including digits) enclosed within double quotes (" ").

If the string literal contains the escape character '\', the following three characters must be digits that specify an entry in the ASCII table in a similar manner to character literals. The exception to this is that the character following the escape character can be an entry from the table above of special escape characters (i.e 'n', 'r', '\' or '0') which will be interpreted according to the numeric association presented in the table. The regex for string literals is ([^ '\\' '"' '\n'] | ('\\'[^ '\n']))*

The following are all valid examples of string literals:

```
“Hello, World!”

“Hello,\040World!”

“C:\\\\Users”

“Each\nword\not\na\nnew\nline\n”
```

Any string literal is typed as the built-in type string (discussed later), and escaped characters are sequences as their ASCII equivalents.

1.1.4 Operators

Below is the list of operators available in the C-net programming language:

```
// Arithmetic: + - * / %

// Assignment: = += -=

// Relational: > < >= <= == !=

// Logical: && || ! ~

// Expression: [] .
```

Note: the index operator ([]) needs an integer expression in between the two square brackets.

1.1.5 Delimiters

The final types of tokens in the C-net language are broadly classified as delimiters. These can be either one of

1) **Comments** : A string of characters beginning with /* and ending with */, **OR** A string of character beginning with // and ending with a new line character; For example,

```
// Here we are using the double backslash

// for single line comments

/*  but here we are using the forward slash

     and star

     for a multiline comment */
```

2) **Whitespace**: Any whitespace character such as a new line, a horizontal tab or a carriage return.
3) Opening and closing **curly braces** "{" and "}"
4) Opening and closing **parentheses** "(" and ")"
5) **Commas** ","
6) **Semicolon** character ";

- Comments and whitespace are used to separate tokens in program source code, and are otherwise insignificant to the program.
- Opening and closing curly braces, on the other hand, have significance in delimiting scope, declaration of user-defined structs and functions which are all discussed in more detail later on.
- Opening and closing parentheses can delimit function arguments and can be used in arithmetic expressions.
- Commas can be used to delimit array elements and function arguments.
- Similarly, a semicolon is used to delimit statements.

2. Types

In the C-net programming language, all identifiers, literals, and expressions must have a type associated with them. All operations have a predefined set of types that they operate on and return, and the compiler throws an error if the user attempts to use an operator on identifiers, expressions or literals that are typed differently than what the operator is defined to handle. For this reason, **strong type checking is an essential feature of the C-net programming language.** In general, the types in C-net can be classified as primitive data types and complex data types.

2.1 Primitive data types

Primitive data types are all numeric types that have pre-defined properties and representations in memory. In addition, they are somewhat special in that the data they represent is stored on the stack at program runtime. The following are the primitive data types defined in the C-net language:

1. **int**

The int type stores numeric values in 4 bytes of memory. It is the type given to any integer literal or an identifier that is typed as an `int`.

2. **char**

The char type represents a character by its numeric entry in the ASCII table stored in 1 byte of memory. It is the type of a character literal, an identifier marked as type `char` and operations that return characters such as indexing into a string.

3. **float**

The float type stores a double-precision floating point number in 8 byte (64 bits) of memory, following the standard of double precision representation. It is the type of a floating point literal or an identifier marked as `float`, as well as operations that return float types.

3.2 Complex data types

There are several built in complex data types in C-net, and the user is allowed to create their own using structs. What makes complex data types different in C-net is that the data they represent resides in heap memory at runtime. An identifier typed as a complex data type is a 64 bit sized reference to a location in memory (a pointer). The specific complex data type determines how the memory that it references is to be manipulated by operators and what it is referencing.

The following are the complex data types that are allowed in C-net.

1. **struct** *`struct_name`*

The struct type is a collection of primitive data types and complex data types that are represented together in memory.

A struct must be given a name when it is declared by the user. Along with its name, the user must specify the elements of a struct along with their corresponding type and names. These elements will be called the members or fields of the struct. In the following example, a new type called "struct person" is declared. It has the two fields shown below: .

```
// Name and age are the members of the struct person.

struct person{
  string name;
  int age;
};
```

A member of a struct may also be typed as a struct, including a struct of type itself. This means that the user is able to nest structs in C-net given that the nested struct is defined before the nesting struct.

Example:

```
struct person{ // Person is defined before couple
  string name;
  int age;
  struct person partner;
};
struct couple{
 struct person p1;
 struct person p2;
};
```

After it is defined by the user, any identifier declared to have a type of struct will become a reference to a location in memory. The location in memory will be the size of all the combined sizes of its members.

Members of a struct can be referenced using the . operator (discussed below in more detail).

Note: If a copy of an id is made, in the return of a function call for example, the copy will refer to the same struct in memory as the original, which is in contrast to strings. E.g.,

```
struct person{
  int age;
};
struct person f(struct person x){
  struct person a = x;
  return a;
}
int main(){
  struct person x = new struct person;
  x.age = 20;
  struct person b = new struct person;
  b = f(x);
  stdout.writeln(soi(b.age)); //  prints 20
  x.age = 15;
  stdout.writeln(soi(b.age)); //  prints 15
  return 0;
}
```

2. string

The opaque string type represents a sequence of characters, and is the type assigned to a string literal or an identifier marked as `string`, as well as the result of some expressions and operations. The underlying implementation of strings are structs and users do not have access to the string struct members. Implementation for string:

```
struct string { void (*cnet_free) (void *); char *data; int length;}
```

Though its implementation is hidden from the programmer, certain attributes of the string can be retrieved/modified through member *functions*. These member functions are accessed through the (.) operator just like struct fields, but they are function calls and must be treated as such.

Immutability of strings

C-net strings are immutable meaning that once a string is created, a user cannot change the string. Any member functions that change the string, actually return a new string object. E.g., capitalization function on a string returns a new capitalized string object. Other member functions do not create new strings as they don't modify the string. E.g., string length function and more (see below).

3. socket / file

Sockets and files are a complex data type that represents a specific network socket or an open file associated with the running program, respectively. Though they are different types, the interfaces provided to the user for operating on them are almost identical.

Much like the string type, their representation is struct-like, providing access to attributes the user may want—such as port number or protocol for sockets and filename and protection for files—through functions that can be called on those objects. The implementation of socket and files is as follows

```
struct cnet_socket {
      void (*cnet_free) (void *sock); FILE *f; int io_type;
      int fd; int port; int type;
      struct sockaddr_in *addr;
};
```

Two transport layer network protocols are supported for sockets by C-net, the **UDP and TCP protocols.** See Appendix for code examples.

4. array

An array is a collection of a number of items of the same type. The type of an array is based on the objects that it contains. For example, an array containing ints would be an int array.

An array can contain either primitive or complex data types. The user must specify how many of the type's objects an array will contain and the memory allocation will be done at runtime according to how much a single object of that type requires and how many of that type an array contains.

Once created, an array is fixed in size. However, **the contents of the array are mutable,** and references to arrays passed around through functions all refer to the same object in memory. E.g.,

```
int main() {

    // Declare size three array and then set its members

    int[] a = new int[3];

    x[0] = 1;  // initialize members

    x[1] = 2;

    x[2] = 3;

    // Declare and set members of size three array simultaneously

    int[] b = new int[3]{1,2,3};

   // Syntax error: arguments must be passed in or use new int[5] without {}

   int[] c = new int[5]{};

}
```

3. Operators

3.1 Operator Types

3.1.1 Arithmetic operators

Arithmetic operators are binary and are left-to-right-associative. The *, /, and % have the same precedence which is higher than the precedence of + and -. + and - have the same precedence.

The following table summarizes the types of operands that operators are defined on, what the operation returns, and what the return value is

Arithmetic operator	Types on which operator is defined		Return type of operator	The output of the operation on the specified operands
+	int	int	int	Integer addition
	float	float	float	Floating point addition
	char	char	char	Returns the character represented by the sum of the ASCII representation of the two characters

	string	string	string	String concatenation
-	int	int	int	Integer subtraction
	float	float	float	Floating point subtraction
	char	char	char	Returns the character represented by the difference of the ASCII representation of the two characters
*	int	int	int	Integer multiplication
	float	float	float	Floating point multiplication
	string	int	string	String concatenation the given integer number of times
%	int / char	int / char	int	The remainder after integer division whose sign is the same as the dividend

3.1.2 Relational operators

The relational operators are <, >, <=, >=, ==, and != and are left-to-right-associative. <, >, <=, and >= have the same precedence, ==, and != have relatively lower precedence. **All relational operators have lower precedence than arithmetic operators.**

In addition, == and != are defined for strings and compares if the characters contained by two strings are the same. E.g.,

```
int main() {

    string a = "abc";

    string b = "abc";

    a == b; // value of this expression is 1

    a != b;  // value of this expression is 0

}
```

3.1.3 Logical operators

The logical operators take one or two operands and return a value based on some logical operation. They have lower precedence than the relational operators.

The following table summarizes the logical operators available in C-net. Logical operators are defined on `int` and `char` types and they return an `int` type.

Operator	The output of the operation on the specified operands
~	An integer resulting from the bitwise not of all the 32 bits in the original integer
!	A 1 if the operand is 0 and a 0 otherwise
&&	A 1 if both the operands are non-zero and a 0 otherwise
||	A 1 if both either one or both of the operands are non-zero and a 0 otherwise

3.1.4 Expression operators

Expression operators are '[]' and '.', they are left-to-right-associative, and have the highest precedence.

The '[]' is defined differently based on the type that it is operating on. *type* is used to show that the array can be of any type, and the return will be of the same type. For example, the return value of an index into a char array will be a char.

Operator	Types on which operator is defined		Return type of operator	The output of the operation on the specified operands
[*x*]	*type* array	int	*type*	The element at the index *x*
	string	int	char	The *x*-th character in the input string

The '.' operator is defined in user defined structs as well as the built in string and socket types, although the behavior is different for the two as discussed in the types section above.

Operator	Types on which operator is defined	Return type of operator	The output of the operation on the specified operands
.name	*struct*	Type of *name*	The value of the field named *name* in the struct the operator is applied on. The compiler throws an error if the struct does not contain a field called *name*
.name()	string, socket	Return type of *name*	Returns the result of invoking the built-in function *name* on the object the operation is performed on

3.1.5 Unary operators

The unary operator (-) assigns negates an int and has higher precedence than the arithmetic operators and is lower in precedence to the expression operators. The not operator (!), returns 1 if the int it is applied to is non-zero and zero otherwise.

3.1.6 Assignment Operators

The assignment operators are = -= and +=

Operator	Behavior

=	assigns an expression on the right hand side of the = to a variable or struct member on the left hand side of the =
+=	adds the expression to the right of it to the variable to the left of it and assigns the result to the variable. *var* += *expr* is semantically equivalent to *var* = *val* + *expr*
-=	Subtracts the expression to the right from the variable on the left and assigns the result to the variable. *var* -= *expr* is semantically equivalent to *var* = *val* - *expr*.

Assignment operators are right-to-left-associative, have the same precedence level to one another, and have lower precedence than the logical operators.

3.2 Operator Precedence & Associativity

C-nets operator's precedence and associativity is listed in the table below in descending precedence order:

Operator	**Symbol**	**Associativity**
Dot	.	Left
Not	~	Right
Times, Divide	*, /	Left
Plus, Minus	+, -	Left
Relational Operators	> , < , >= , <=	Left
Equality operators	==, =!	Left
And	&&	Left
Or	||	Left
Assignment	=	Left

4. Expressions

An expression is a combination of operators, constants and variables. It is a value held in a memory location and can appear on the right hand side of an assignment operator. An expression may consist of one or more operands, and zero or more operators to produce a value. Example:

```
x + y

10.75

a = 4
```

The different types of expressions in C-net can be classified in the following manner.

4.1.1 ids

An id, itself being an expression, is one that represents a location in memory at runtime and can appear on the left side of an assignment operator as a/an:

- Identifier: the name of the variable of any type can be used as an identifier and be on the left hand side of an assignment to an expression of the same type. The regex is

  (['a'-'z' 'A'-'Z'] | '_') (['a'-'z' 'A'-'Z'] | ['0'-'9'] | '_')*

- Members of a struct type. For example, if we have the following struct,

  ```
  struct Person {
    string name;
    int age;
  };
  ```

  And we had a person struct as follows:

  ```
  struct person p1 = new struct person;
  ```

  p1.name is a valid lvalue that can be assigned an expression of type string

- An index into an array can appear on the left side of an assignment. E.g.,

```
int[] a = new int[1];
x[0] = 1;  // x[0] is a location in memory
```

4.1.2 Literal expressions

```
35.5 // float literal expression
'a'  // character literal expression
"goat" // string literal expression
35 // integer literal expression
```

These evaluate to and return their representation as discussed earlier in the manual.

4.2.2 Operation expressions

These are expressions that are obtained by applying one of the available operators on an expression or pair of expressions. The return value is the result of the expression and is also an expression. Possible examples include

```
a = b // assignment operator; returns a after assignment
x & 4 // and operator
y = (a + (b = 9)) // multiple operators; returns y
```

4.2.3 New expressions

A new expression is the keyword new followed by either

1. *struct struct_name* where *struct_name* is a valid struct OR
2. *type[]* where type is a valid type as discussed above OR
3. *type[]{val1, val2, val3….}* where val[n] are all expressions of type *type*

Example:

```
new struct person[] // type : struct person array
new int[5]{1, 5, 9, 3, 12}  // type : int array
```

The new expression allocates memory for the object that is getting created and returns a reference to it. As such the type of a new expression is always the type of the object that is requested.

4.2.4 Function calls

A function call is a function name followed by the arguments to be passed to the function separated by commas and enclosed with parentheses. The arguments passed to the function must match the function's signature both by number and type (functions discussed later). An example function call might look like:

```
myfunc(5,2) // function must accept two arguments typed int
```

5. Declarations

Declarations are a class of statements in C-net that bind a certain identifier with a type and possibly a value. The syntax of declarations is that a type is specified followed by an identifier and a semicolon. In general, C-net allows mixing pure declarations with declarations and assignments by preceding the semicolon with an expression which has the same type as the identifier being declared. The exception to this are structs (see below).

Example:

```
int x; // valid -- declares a variable x of type int
int y = 5; // also valid -- declares y and assigns 5 to it
```

In general, any type can be declared and optionally assigned to an expression of its own type as in the following examples:

```
int[] x = new int[15]; // 15 is the size of the array

struct person p1 = new struct person; //declare a struct person
```

```
Struct person p1; //not a valid declaration

p1.name = "Joe"; // assign the member name to "Joe"

p1.age = 40; // assign the member age to 40
```

Arrays can be populated in either of the following two ways.

1. by a sequence of statements indexing the array by position and assigning it to a given value after the array has been initialized using a new expression

```
int[] x = new int[10];

x[0]= 1;

x[1]= 2; ...
```

2. by following the array declaration with the elements enclosed in curly braces

```
int[] x = new int[5]{1,2,3,4}; // first 4 elements initialized
```

5.1 Sockets and files

Objects of the socket/file types can be declared like any other variable, and they can be created using the built-in functions for creating sockets/files.

5.1.1 Sockets

Sockets in C-net are created using the built-in *nopen* function. It's signature is:

```
socket nopen(string address, int port, string protocol, string type)
```

E.g.

```
socket sock = nopen("www.google.com", 80, "tcp", "connect");
```

5.2.4 Functions

A function declaration in C-net is the return type of the function, followed by the name of the function and an optional parameter list of comma separated parameters enclosed in parentheses:

```
type identifier(optional parameters)
```

This is the signature of the function and must immediately be followed by its definition, which is an open curly brace, a collection of 0 or more statements, and a close curly brace:

```
type identifier(optional parameters){optional statements}
```

A function that has a type other than void is required to have a return statement at the end of the function. Return statements are discussed below in the statements section.

E.g.

```
int multiply(int x, int y) // this is the function's signature
{
      return x * y; // a return statement required
}
```

6. Statements

Statements are a sequence of C-net code ending in a semicolon followed by any number of statements.Unlike expressions, statements inherently don't have a value .

6.1 Expression statement

An expression statement is an expression followed by a semicolon (i.e., "expr;").

```
 5;      2 + 2;       X++;        y = x + 25;
```

6.2 Declaration statement

Declarations (discussed above in detail) are also a subclass of statements.

6.3 Control flow

6.3.1 if/else

The forms of conditional statements are the following

```
if ( expression ) statement

if ( expression ) statement else statement
```

The expression is evaluated and if it is a non-zero, the statement following the expression is executed. If the user utilizes the second case, when the expression evaluates to a zero, the second sub-statement is implemented.

6.3.2 Loops

while loop

The while loop has the following form:

```
while ( expression ) statement
```

This means that the statement is executed repeatedly in a loop as long as the expression doesn't evaluate to zero. The expression is evaluated before each iteration of the loop.

for loop

The for loop statement has the following form:

```
for ( expression-1opt ; expression-2opt ; expression-3opt ) statement
```

Expression-1 represents the initialization for the loop. The expression-2 is evaluated before each iteration and if it evaluates to a non-zero, expression-3 is executed. The loop is exited when expression-2 returns a zero.

6.3.3 break and continue

- The break statement terminates the flow in a while or for statement. The control after termination is passed to the statement following the terminated statement.
- The continue statement skips the current iteration of the loop and continues with the next iteration.

```
break ;

continue ;
```

6.4 Block Statements

Block statement is a series of one or more statements enclosed by an opening and closing curly braces ({}). Block identifies the scope of the primitive type variables that are declared in it. Primitives are stored on the stack by default. Therefore, any primitive type declared within a block can't be accessed outside of the block statement. This is because the stack is rolled right before the closing curly brace removing all access to the primitive type data stored on the stack. In contrast, non-primitives are stored on the heap by default. Therefore, the user has to explicitly delete the identifier associated with the non-primitive type data within the given block in order to avoid a memory leak.

6.5 Return Statements

A function returns to its caller by means of the return statement. The return statement my return with no value as in the case of `return ;` or the statement might return with the value of an expression that is supplied to it in the form of `return (expression) ;`

7. function

A function is a series of statements that can be executed repeatedly with possibly varying inputs. A function has a type which includes the return type of the function and the number and type of inputs that the function is expecting. Example of a function signature and definition defined above:

E.g., the signature of a function named echo that takes in one argument

```
string echo(string s)
```

E.g., the definition of echo; it returns its argument

```
string echo(string s){return s}
```

Note: The body of the function would have to follow this declaration immediately as C-net does not allow separate function declaration and definition.

8. Built-in functions

The list of functions available for the built-in string and socket types are given below, along with their return types. They are accessed the same as struct members and called the same as functions. If we have a string variable named *`str`*, *`str.length()`* is the way to call the length function.

String:

Function signature	**What the function does**
int length()	Returns the length of the string it is called on
string upper()	Returns the string it is called on in upper case (only the alphabetic characters change)
string lower()	Returns the string it is called on in lowercase
int atoi()	Returns the integer represented by the string. If it is not a valid integer, it will return a 0
float atof()	Returns the float represented by the string
int find(char c)	Searches for c in the given string a and returns the index of the first occurrence
string substring(int b, int e)	Returns a substring of the string from index b inclusive to index e exclusive
String reverse()	Returns the reverse of the string it is called on

Socket:

Function signature	What the function does
string read_line(int max_length)	Reads from the socket a maximum of max_length characters or until a new line is read and returns a string which holds the content that was read.
int print_line(string s)	Sends the string s followed by a new line character into the socket it is operating on and returns the number of bytes successfully written.
string read(int max_length)	Reads up to a max_length number of bytes from the socket and returns a string containing that data
int write(string s)	Writes the string s with no new line character and returns the number of bytes successfully written

4. Project Plan

Planning

To complete the C-net language and compiler in a timely manner, we had team meetings twice a week since the start of the project. In addition, we had a third meeting with our TA every week to update him on our progress and ask for feedback on the changes we had made for that given week. Our TA meetings had to be adjusted after our assigned TA left the course near the end of the semester. At which point, we met with our new TA to ask for his advice on how to approach the final steps of the project.

Our team communication took place over Slack. We shared meeting docs and drafts of documents through a shared google drive folder. Our communications were effective and efficient because we made sure to update each other during each step of the project.

Specification

The goal of C-net was to be compatible with C. We implemented some of the basic elements of C such as operators, identifiers, and functions. We built on this basic skeleton

structure a language that creates a seamless way for users to implement network/file programming through succinct code and easy interaction with files and sockets. C-net does this by providing an abstract wrapper for files and network sockets as objects for reading and writing along with built-in methods for performing common manipulations.

Development

At the beginning of the project, before the submission of the language reference manual, our meetings were focused on designing and discussing the high-level detail of our language. After the LRM, we built the scanner, parser and AST respectively. For the Hello World assignment, we built a very simple sast and code gen. We spent the final weeks of the semester testing, updating and expanding our sast and code gen respectively.

Testing

Features were tested before being merged with the main branch of our github repository. Tests featured both success and fail cases. These tests were then added to the testing suite that would automatically be tested when the C-net compiler was created.

Software Development Tools

We used the following programming and development environments when creating C-net:

- **Libraries and Languages**: Ocaml Version 4.11.1including Ocamlyacc and Ocammllex and LLVM Version 11.1.0 was used.
- **Software**: Development was done in vim, and Visual Studio Code.
- **OS**: Development was done on OSX 10.15 and Ubuntu 18.04.1.
- **Version Control**: Github-Hosted Git Repository

Roles and Responsibilities

Team Members	**Responsibility**
Rediet Bekele	Test Suite, Final Report
Kidus Mulu	Semant, Code gen, Scanner, Parser
Kingsley Neequaye	Test Suite, Final Report
William Oseghare	Code gen, standard library
Bruk Zewdie	Semant, Code gen, Scanner, Parser

Project Timeline

Time	**Deadline**
January 21	First group meeting to work on the project proposal
February 2	Finalized changes to the proposal
February 3	Proposal submission
February 11	First meeting with our assigned TA
February 20	Finalized the Scanner and Parser
February 24	Finalized LRM & Scanner, Parser and LRM Submission
March 13	Semant and codegen for hello world
March 24	Hello World program submission
April 20	Finalized codegen and Semant for the language
April 23	Final changes to project files
April 26	Report Due
April 26	Presentation by the group

Project Log

See Appendix for github log and commit messages for our project.

Style Guide for Programming

We had five contributors to the github repository for our project so we made sure to follow strict style guidelines for programming and committing in order to avoid any conflicts. We followed a simple workflow where each person created a feature branch and made pull requests to master. We used this simple git collaboration workflow (Simple Git workflow for collaborating on a project. I wrote this to help a co-worker learn Git (and help me remember after a year of working on my own).) for all the git commands that we made.

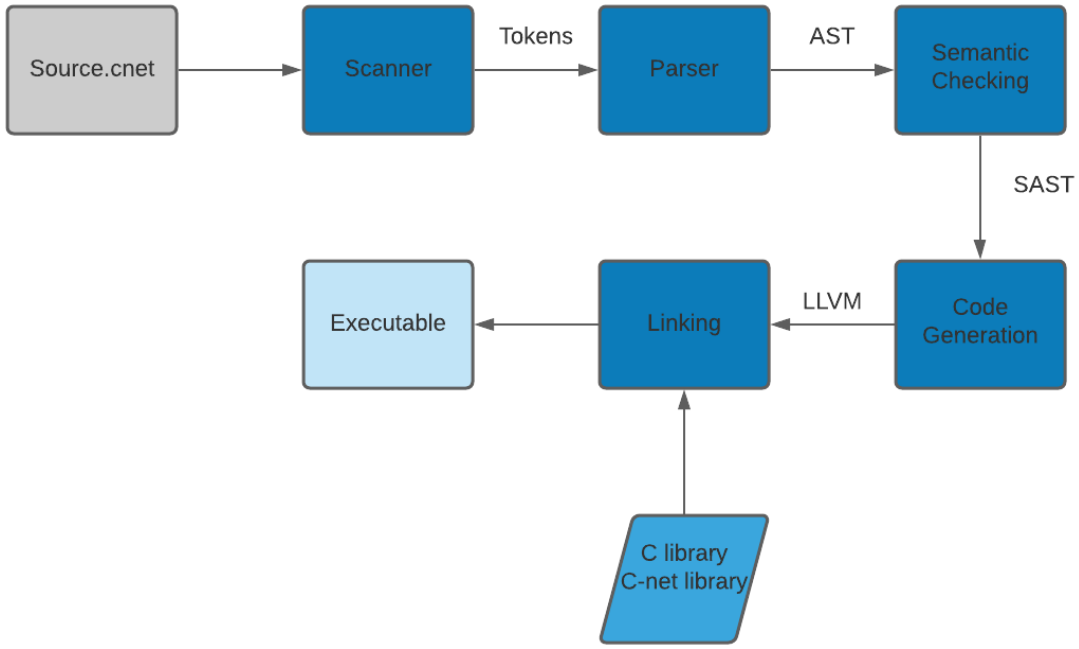
We also made sure to create temporary feature branches for implementing small but working features. Once we had implemented this task and our pull request had been merged to master, we made sure to delete the current feature branch and create a new one for our next feature. When we made a pull request, a github CI action immediately started running and checked the code for any fails or merge conflicts.

We noted that having standardized commit messages that are easy to read and understand would make the programming workflow smooth and easy to implement. We referred to *Seven Rules of a Great Commit Message* (https://chris.beams.io/posts/git-commit/#separate) with special focus on the second, third and fifth rules.

When writing code, we decided that file and function names should be as descriptive as possible to avoid confusion and to maximize readability. We also stuck to a maximum of 80 character lines for readability.

5. Architectural Design

Block Diagram



Scanner

The scanner (scanner.mll) takes in a C-net input program as input and produces as an output a stream of tokens. The scanner uses syntactic rules, which specify how to classify characters into tokens for the C-net source language. Blank space (spaces, tabs, and end-of-line characters) and comments are ignored. If any characters are syntactically invalid, the scanner throws an error.

Tokens for valid characters are used by the parser in the next stage of compilation. Kidus and Bruk implemented the Scanner.

Parser

The parser (parser.mly) takes in the tokens produced by the scanner and generates an AST (abstract syntax tree), following the syntactic model of the C-net source language (formally, a context-free grammar). The AST is used in later phases of compilation. If the input stream of tokens is not a valid C-net program, the parser throws an error. Kidus and Bruk implemented the parser.

Semantic Checking

The semantic checker (semant.ml) takes in the AST and newly defined types listed in sast.ml to output a semantically checked abstract syntax tree (SAST). The semantic checker goes beyond syntax to do strong type checking consistent with C-net's type system and context sensitive analysis, ensuring scoping rules are followed. It also checks that operators are used with the correct types, global variables are tracked and scoped appropriately, there are no invalid type assignments, function calls are valid (correct number and type of passed in arguments), and that structs can be nested and accessed giving the correct type. If there are type or scope errors, diagnostic messages are outputted. Kidus and Bruk implemented the semantic checker.

Code Generation

The codegen (codegen.ml) takes the SAST and produces LLVM IR (intermediate representation). In this phase, LLVM code is generated for all C-net source language constructs (types, structs, functions, etc.). Kidus, Bruk, and Will implemented codegen.

C libraries

We implemented a C-net standard library (libcent/) using C's standard library. Hence, we linked in the C libraries. Will implemented the C-net standard library in C.

6. Test Plan

We have several tests located in the tests/ directory. We broke tests down by topic into subdirectories: scanner/, parser/, semant/, stdlib/, and integration/. Test files with the name test-*.cnet are expected to pass and fail-*.cnet are expected to fail, and the output for each test type goes into *.out and *.err files, respectively. Our testing suite and automation is inspired by the Micro-C testing suite and uses a test script (runtests.sh), which runs all of the tests in tests/ each time a program is compiled using make. This ensures that any changes made did not introduce any errors to previously functioning features of our compiler. Rediet and Kingsley worked on testing.

Unit Testing

As we were developing the different layers of the compiler, we would create tests to test newly added features. Thus, we unit tested the scanner, parser and semantic checker as they were being built, and once the bulk of features were finished and tested for a particular compiler phase, we would move on to implementing and testing the next phase of the compiler.

Integration tests

Once we implemented the codegen for C-net, we worked on integration tests. These tests follow the same test file naming convention and behavior mentioned above. These tests were based on actual programs a user of the C-net language may write or to test specific programs of diagnostic interest. See section 8 for representative C-net source language programs and their target LLVM output programs.

7. Lessons Learned

Rediet

It really helps to have frequent meetings with your teammates to discuss updates. Also, make use of office hours with Professor Edwards and the TAs to get help with your proposal. Of course, you will end up making significant changes to the proposal as the semester goes on. However, discussing your higher-level language idea with the teaching staff makes it easier for you to narrow down your scope and focus on a succinct project when you start implementing code. Furthermore, don't get bogged down in the details. Instead of trying to implement every feature in your sast and then generating a codegen, make sure that you have a simple and working sast and codegen before adding more features to your language. Lastly, start early! You might think that you are ahead of schedule but it only takes one bug to set you back weeks. So make sure to start the project as soon as it is given out.

Kidus

Through this project I would say I have learned three important things. The first one is the design and architecture of a compiler as a whole. As such, I would encourage future teams to think about the entire thing in perspective, from lexing all the way to code generation, when they do the project. The second important thing that I have enjoyed a lot is that I was introduced (for the first time) to OCaml and functional programming as a whole. Before PLT, my only programming experience was in C and other C-like languages, so it was quite a transition when I first started programming in OCaml. Now, at the end of the semester, I would say that OCaml is high among my favorite programming languages. In particular, there are three things that I really liked in OCaml. First is the level of rigor in the type checking which I found took off a lot of complexity in thinking about the compiler. Many times, simply changing around things in the code so that the types matched led to the correct code with the desired behavior. The second is OCaml's warnings on unmatched cases that is a very helpful debugging tool even before the compiler is tested with actual programs. Finally, the third is the fact that nothing has any side effects, which I thought led to a very natural expression of what the programmer wants to happen. Lastly, through working on our compiler, I had a very pleasant experience of working on a team. My main advice in that regard would be to always keep in mind that the entire team has the same goal when doing the project, and when working as a team member, that should be your only goal.

Kingsley

One of the most important lessons I learned is that when coordinating with a team communication is key. Having frequent meetings to catch up on team members' workflow helps

keep everyone on the same page. Another important lesson that I learned is that breaking down the problem into smaller working chunks helps make the task of writing a compiler less daunting. Looking back, having unit tests for the scanner, parser, and semantic checker helped to get one part of the compiler working well enough to implement the subsequent phase of compilation. This kind of workflow I think is more efficient/agile and I will take it with me and apply it to future projects. As for advice for future teams, I would second the common phrase "start early!" In addition to that, I would say that choosing the appropriate teammates is very important to having a good working environment, and how well you work with your team is indirectly related with how painful the project will be. Lastly, I would suggest thinking through how everything works from the compiler architecture perspective down the weeds of your implementation to better appreciate how everything fits together and what you and your team have built.

William

For one I learned that when you have a really committed team, most of what Edwards warns you about in the first class doesn't really apply. More importantly, I learned a lot about the importance of good design and modularization. We were able to properly modularize the entire project and has a test suite dedicated to each of those modules. It seemed like a lot to do the workflow setup at the beginning, but it only saved us tons of productive time in the ended. The workflow enabled us to spend most of our time working on the actual project. All in all, it's really important to have teammates that understand workflow and good design. Saves from a lot of unnecessary conflict ahead (code and personal).

Bruk

Iterative Design: We had to rethink the design of our programming language multiple times at different stages of the project. Initially, it was hard to come up with a good design of the project proposal mostly because we didn't know how feasible some of our ideas were. But as we learned more about compiler design, we were able to have a clear picture of what we wanted to implement in our programming language.

Testing: Don't procrastinate on writing tests until you have something "testable". Having a suite of regression tests for each moving part (scanner, parser, semantic checker, custom library) allowed us to very easily pinpoint new bugs in our code.

Continuous Integration: We used **github actions**, a continuous integration tool provided by github to automate our integration and unit tests. Running the tests remotely before any of our pull requests got merged allowed us to prevent any non-working code from being committed to master. This saved us a whole lot of time and headache we would have gone through in order to figure out which commit is responsible for a certain breaking change. Setting up the workflow is also not that hard, so I highly suggest doing it at the beginning of your project.

8. Example test programs

Example Code 1: http-server.cnet

```
int main(string[] argv)
{
      /* language provided array has built in length and safe member access */
      if (argv.alength() != 3){
            /* stdout/stdin is just another file */
```

```
            stdout.writeln("usage: " + argv[0] + " <server-port>
<web-root>");
            return -1;
      }

      string port = argv[1];
      string webroot = argv[2];

      /* simple socket creation/deletion */
      // Create a listening socket (also called server socket)
      socket listener = nopen("", port.toint(), "tcp", "listen");
      stdout.writeln("Listening on port: " + port);

      while (1) {
            socket clntsock = listener.naccept(); /* gives connected socket
*/
            /* files/sockets have similar well-defined read/write interfaces
*/
            string req_line = clntsock.readln();

            // Accept an incoming connection
            string[] tokens = new string[3]{"", "", ""};
            req_line.split(" ", tokens); /* standard library operations on
strings are very convenient */

            /* never have to worry about string management, everything is
automatically cleaned up */
            string method = tokens[0];
            string req_URI = tokens[1];
            string httpVersion = tokens[2];
            string fileName;
            string respHeader;

            if (method.slength() == 0 || httpVersion.slength() == 0 ){
                  respHeader = "501 Not Implemented";
            } else if (method == "GET") {
                  respHeader = "200 OK";
                  fileName += webroot;
                  fileName += req_URI;
            } else {
                  stdout.writeln("unsupported HTTP method: " + method);
                  return -1;
            }

            // if uri ends with a '/', append index.html
            if (req_URI.charat(req_URI.slength() - 1) == '/'){
                  fileName += "index.html";
            }

            file targetFile;
            // Try to open the file or give a 404
            targetFile = fopen(fileName, "rb");
            if (targetFile.error()){
                  respHeader = "404 Not Found";
```

```
                  stdout.writeln("Requested file " + fileName + " could not
be found");
            } else {
                  respHeader = "200 OK";
            }
            // log the request
            stdout.writeln(req_line);

            if (respHeader == "200 OK")
                  clntsock.writeln("HTTP/1.0 200 OK\n\n" +
targetFile.readall());
            else
                  clntsock.writeln("HTTP/1.0 404 Not Found\n\n");

            stdout.writeln("Finished with current client");

            delete clntsock;

      }
      return 0;
}
```

LLVM Output 1: http-server.cnet

```
; ModuleID = 'CNet'
source_filename = "CNet"

%cnet_file = type { %string*, %string*, i32 }
%string = type { %string*, %string*, i32 }
%array = type { %string*, %string*, i32, i32 }

@cnet_stdin = external externally_initialized global %cnet_file*
@cnet_stdout = external externally_initialized global %cnet_file*
@tmp = private unnamed_addr constant [8 x i8] c"usage: \00", align 1
@tmp.1 = private unnamed_addr constant [26 x i8] c" <server-port>
<web-root>\00", align 1
@tmp.2 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.3 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.4 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.5 = private unnamed_addr constant [4 x i8] c"tcp\00", align 1
@tmp.6 = private unnamed_addr constant [7 x i8] c"listen\00", align 1
@tmp.7 = private unnamed_addr constant [20 x i8] c"Listening on port: \00",
align 1
@tmp.8 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.9 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.10 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.11 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.12 = private unnamed_addr constant [2 x i8] c" \00", align 1
@tmp.13 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
```

```
@tmp.14 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.15 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.16 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.17 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.18 = private unnamed_addr constant [4 x i8] c"GET\00", align 1
@tmp.19 = private unnamed_addr constant [20 x i8] c"501 Not Implemented\00",
align 1
@tmp.20 = private unnamed_addr constant [7 x i8] c"200 OK\00", align 1
@tmp.21 = private unnamed_addr constant [26 x i8] c"unsupported HTTP method:
\00", align 1
@tmp.22 = private unnamed_addr constant [11 x i8] c"index.html\00", align 1
@tmp.23 = private unnamed_addr constant [3 x i8] c"rb\00", align 1
@tmp.24 = private unnamed_addr constant [14 x i8] c"404 Not Found\00", align 1
@tmp.25 = private unnamed_addr constant [16 x i8] c"Requested file \00", align
1
@tmp.26 = private unnamed_addr constant [20 x i8] c" could not be found\00",
align 1
@tmp.27 = private unnamed_addr constant [7 x i8] c"200 OK\00", align 1
@tmp.28 = private unnamed_addr constant [7 x i8] c"200 OK\00", align 1
@tmp.29 = private unnamed_addr constant [18 x i8] c"HTTP/1.0 200 OK\0A\0A\00",
align 1
@tmp.30 = private unnamed_addr constant [25 x i8] c"HTTP/1.0 404 Not
Found\0A\0A\00", align 1
@tmp.31 = private unnamed_addr constant [29 x i8] c"Finished with current
client\00", align 1

declare i32 @cnet_free(%cnet_file*)

declare i32 @alength(%array*)

declare i8 @charat(%string*, i32)

declare void @split(%string*, %string*, %array*)

declare i8 @find_char(%string*, i8)

declare %string* @reverse(%string*)

declare %string* @substring(%string*, i32, i32)

declare %string* @lower(%string*)

declare %string* @upper(%string*)

declare %string* @user_soi(i32)

declare i32 @toint(%string*)

declare double @tofloat(%string*)

declare i32 @slength(%string*)

declare i32 @cnet_strcmp(%string*, %string*)
```

```
declare %string* @cnet_strcat(%string*, %string*)

declare %string* @cnet_strmult(%string*, i32)

declare %string* @cnet_strcpy(%string*, %string*)

declare i32 @error(%cnet_file*)

declare %string* @readall(%cnet_file*)

declare %string* @readln(%cnet_file*)

declare i32 @nwrite(%cnet_file*, %string*, i32)

declare i32 @writeln(%cnet_file*, %string*)

declare %cnet_file* @user_fopen(%string*, %string*)

declare %string* @read(%cnet_file*, i32)

declare i32 @write(%cnet_file*, %string*)

declare %cnet_file* @naccept(%cnet_file*)

declare %cnet_file* @user_nopen(%string*, i32, %string*, %string*)

declare %string* @cnet_new_str_nolen(i8*)

declare %string* @cnet_empty_str()

declare i8 @cnet_char_at(%string*, i32)

declare i8* @memset(i8*, i32, i64)

declare i32 @main(i32, i8**)

define i32 @user_main(%array* %argv) {
entry:
  %argv1 = alloca %array*
  store %array* %argv, %array** %argv1
  %argv2 = load %array*, %array** %argv1
  %alength_result = call i32 @alength(%array* %argv2)
  %tmp = icmp ne i32 %alength_result, 3
  %tmp_cast = sext i1 %tmp to i32
  %tmp3 = icmp ne i32 %tmp_cast, 0
  br i1 %tmp3, label %if_body, label %elif

if_body:                                          ; preds = %entry
  %tmp1000 = alloca %string*
  %strlit = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([8 x
i8], [8 x i8]* @tmp, i32 0, i32 0))
  %arr = load %array*, %array** %argv1
  %idx_elt = call %string** @cnet_index_arr(%array* %arr, i32 0)
  %argv4 = load %string*, %string** %idx_elt
```

```
  %cnet_strcat_result = call %string* @cnet_strcat(%string* %strlit, %string*
%argv4)
  store %string* %cnet_strcat_result, %string** %tmp1000
  %tmp1001 = alloca %string*
  %tmp10005 = load %string*, %string** %tmp1000
  %strlit6 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([26
x i8], [26 x i8]* @tmp.1, i32 0, i32 0))
  %cnet_strcat_result7 = call %string* @cnet_strcat(%string* %tmp10005,
%string* %strlit6)
  store %string* %cnet_strcat_result7, %string** %tmp1001
  %cnet_stdout = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp10018 = load %string*, %string** %tmp1001
  %writeln_result = call i32 @writeln(%cnet_file* %cnet_stdout, %string*
%tmp10018)
  %tmp10009 = load %string*, %string** %tmp1000
  %cast = bitcast %string* %tmp10009 to %cnet_file*
  %tmp10 = call i32 @cnet_free(%cnet_file* %cast)
  %tmp100111 = load %string*, %string** %tmp1001
  %cast12 = bitcast %string* %tmp100111 to %cnet_file*
  %tmp13 = call i32 @cnet_free(%cnet_file* %cast12)
  ret i32 -1

elif:                                             ; preds = %entry
  br i1 true, label %if_body14, label %if_merge

if_body14:                                        ; preds = %elif
  br label %if_merge

if_merge:                                         ; preds = %if_body14
  %port = alloca %string*
  %strlit16 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.2, i32 0, i32 0))
  store %string* %strlit16, %string** %port
  %port17 = load %string*, %string** %port
  %arr18 = load %array*, %array** %argv1
  %idx_elt19 = call %string** @cnet_index_arr(%array* %arr18, i32 1)
  %argv20 = load %string*, %string** %idx_elt19
  %cnet_strcpy_result = call %string* @cnet_strcpy(%string* %port17, %string*
%argv20)
  store %string* %cnet_strcpy_result, %string** %port
  %webroot = alloca %string*
  %strlit21 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.3, i32 0, i32 0))
  store %string* %strlit21, %string** %webroot
  %webroot22 = load %string*, %string** %webroot
  %arr23 = load %array*, %array** %argv1
  %idx_elt24 = call %string** @cnet_index_arr(%array* %arr23, i32 2)
  %argv25 = load %string*, %string** %idx_elt24
  %cnet_strcpy_result26 = call %string* @cnet_strcpy(%string* %webroot22,
%string* %argv25)
  store %string* %cnet_strcpy_result26, %string** %webroot
  %listener = alloca %cnet_file*
  store %cnet_file* null, %cnet_file** %listener
  %strlit27 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
```

```
x i8], [1 x i8]* @tmp.4, i32 0, i32 0))
  %port28 = load %string*, %string** %port
  %toint_result = call i32 @toint(%string* %port28)
  %strlit29 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @tmp.5, i32 0, i32 0))
  %strlit30 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([7
x i8], [7 x i8]* @tmp.6, i32 0, i32 0))
  %user_nopen_result = call %cnet_file* @user_nopen(%string* %strlit27, i32
%toint_result, %string* %strlit29, %string* %strlit30)
  store %cnet_file* %user_nopen_result, %cnet_file** %listener
  %tmp100031 = alloca %string*
  %strlit32 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([20 x i8], [20 x i8]* @tmp.7, i32 0, i32 0))
  %port33 = load %string*, %string** %port
  %cnet_strcat_result34 = call %string* @cnet_strcat(%string* %strlit32,
%string* %port33)
  store %string* %cnet_strcat_result34, %string** %tmp100031
  %cnet_stdout35 = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp100036 = load %string*, %string** %tmp100031
  %writeln_result37 = call i32 @writeln(%cnet_file* %cnet_stdout35, %string*
%tmp100036)
  %tmp100038 = load %string*, %string** %tmp100031
  %cast39 = bitcast %string* %tmp100038 to %cnet_file*
  %tmp40 = call i32 @cnet_free(%cnet_file* %cast39)
  br label %while

while:                                            ; preds = %if_merge219,
%if_merge
  br i1 true, label %while_body, label %merge

while_body:                                       ; preds = %while
  %clntsock = alloca %cnet_file*
  store %cnet_file* null, %cnet_file** %clntsock
  %listener41 = load %cnet_file*, %cnet_file** %listener
  %naccept_result = call %cnet_file* @naccept(%cnet_file* %listener41)
  store %cnet_file* %naccept_result, %cnet_file** %clntsock
  %tmp100042 = alloca %string*
  %clntsock43 = load %cnet_file*, %cnet_file** %clntsock
  %readln_result = call %string* @readln(%cnet_file* %clntsock43)
  store %string* %readln_result, %string** %tmp100042
  %req_line = alloca %string*
  %strlit44 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.8, i32 0, i32 0))
  store %string* %strlit44, %string** %req_line
  %req_line45 = load %string*, %string** %req_line
  %tmp100046 = load %string*, %string** %tmp100042
  %cnet_strcpy_result47 = call %string* @cnet_strcpy(%string* %req_line45,
%string* %tmp100046)
  store %string* %cnet_strcpy_result47, %string** %req_line
  %tmp100048 = load %string*, %string** %tmp100042
  %cast49 = bitcast %string* %tmp100048 to %cnet_file*
  %tmp50 = call i32 @cnet_free(%cnet_file* %cast49)
  %tokens = alloca %array*
  store %array* null, %array** %tokens
```

```
  %strlit51 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.9, i32 0, i32 0))
  %strlit52 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.10, i32 0, i32 0))
  %strlit53 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.11, i32 0, i32 0))
  %cnet_init_array = call %array* (i32, i32, i32, i32, %string*, ...)
@cnet_init_array(i32 8, i32 2, i32 3, i32 3, %string* %strlit51, %string*
%strlit52, %string* %strlit53)
  store %array* %cnet_init_array, %array** %tokens
  %req_line54 = load %string*, %string** %req_line
  %strlit55 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([2
x i8], [2 x i8]* @tmp.12, i32 0, i32 0))
  %tokens56 = load %array*, %array** %tokens
  call void @split(%string* %req_line54, %string* %strlit55, %array*
%tokens56)
  %method = alloca %string*
  %strlit57 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.13, i32 0, i32 0))
  store %string* %strlit57, %string** %method
  %method58 = load %string*, %string** %method
  %arr59 = load %array*, %array** %tokens
  %idx_elt60 = call %string** @cnet_index_arr(%array* %arr59, i32 0)
  %tokens61 = load %string*, %string** %idx_elt60
  %cnet_strcpy_result62 = call %string* @cnet_strcpy(%string* %method58,
%string* %tokens61)
  store %string* %cnet_strcpy_result62, %string** %method
  %req_URI = alloca %string*
  %strlit63 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.14, i32 0, i32 0))
  store %string* %strlit63, %string** %req_URI
  %req_URI64 = load %string*, %string** %req_URI
  %arr65 = load %array*, %array** %tokens
  %idx_elt66 = call %string** @cnet_index_arr(%array* %arr65, i32 1)
  %tokens67 = load %string*, %string** %idx_elt66
  %cnet_strcpy_result68 = call %string* @cnet_strcpy(%string* %req_URI64,
%string* %tokens67)
  store %string* %cnet_strcpy_result68, %string** %req_URI
  %httpVersion = alloca %string*
  %strlit69 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.15, i32 0, i32 0))
  store %string* %strlit69, %string** %httpVersion
  %httpVersion70 = load %string*, %string** %httpVersion
  %arr71 = load %array*, %array** %tokens
  %idx_elt72 = call %string** @cnet_index_arr(%array* %arr71, i32 2)
  %tokens73 = load %string*, %string** %idx_elt72
  %cnet_strcpy_result74 = call %string* @cnet_strcpy(%string* %httpVersion70,
%string* %tokens73)
  store %string* %cnet_strcpy_result74, %string** %httpVersion
  %fileName = alloca %string*
  %strlit75 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.16, i32 0, i32 0))
  store %string* %strlit75, %string** %fileName
  %respHeader = alloca %string*
```

```
  %strlit76 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1
x i8], [1 x i8]* @tmp.17, i32 0, i32 0))
  store %string* %strlit76, %string** %respHeader
  %tmp100077 = alloca i32
  %method78 = load %string*, %string** %method
  %strlit79 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @tmp.18, i32 0, i32 0))
  %cnet_strcmp_result = call i32 @cnet_strcmp(%string* %method78, %string*
%strlit79)
  store i32 %cnet_strcmp_result, i32* %tmp100077
  %method80 = load %string*, %string** %method
  %slength_result = call i32 @slength(%string* %method80)
  %tmp81 = icmp eq i32 %slength_result, 0
  %tmp_cast82 = sext i1 %tmp81 to i32
  %httpVersion83 = load %string*, %string** %httpVersion
  %slength_result84 = call i32 @slength(%string* %httpVersion83)
  %tmp85 = icmp eq i32 %slength_result84, 0
  %tmp_cast86 = sext i1 %tmp85 to i32
  %tmp87 = or i32 %tmp_cast82, %tmp_cast86
  %tmp88 = icmp ne i32 %tmp87, 0
  br i1 %tmp88, label %if_body89, label %elif90

if_body89:                                        ; preds = %while_body
  %respHeader91 = load %string*, %string** %respHeader
  %strlit92 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([20 x i8], [20 x i8]* @tmp.19, i32 0, i32 0))
  %cnet_strcpy_result93 = call %string* @cnet_strcpy(%string* %respHeader91,
%string* %strlit92)
  store %string* %cnet_strcpy_result93, %string** %respHeader
  br label %if_merge133

elif90:                                           ; preds = %while_body
  %tmp100094 = load i32, i32* %tmp100077
  %tmp95 = icmp ne i32 %tmp100094, 0
  br i1 %tmp95, label %if_body96, label %elif97

if_body96:                                        ; preds = %elif90
  %respHeader98 = load %string*, %string** %respHeader
  %strlit99 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([7
x i8], [7 x i8]* @tmp.20, i32 0, i32 0))
  %cnet_strcpy_result100 = call %string* @cnet_strcpy(%string* %respHeader98,
%string* %strlit99)
  store %string* %cnet_strcpy_result100, %string** %respHeader
  %tmp1000101 = alloca %string*
  %fileName102 = load %string*, %string** %fileName
  %webroot103 = load %string*, %string** %webroot
  %cnet_strcat_result104 = call %string* @cnet_strcat(%string* %fileName102,
%string* %webroot103)
  store %string* %cnet_strcat_result104, %string** %tmp1000101
  %fileName105 = load %string*, %string** %fileName
  %tmp1000106 = load %string*, %string** %tmp1000101
  %cnet_strcpy_result107 = call %string* @cnet_strcpy(%string* %fileName105,
%string* %tmp1000106)
  store %string* %cnet_strcpy_result107, %string** %fileName
```

```
  %tmp1000108 = load %string*, %string** %tmp1000101
  %cast109 = bitcast %string* %tmp1000108 to %cnet_file*
  %tmp110 = call i32 @cnet_free(%cnet_file* %cast109)
  %tmp1000111 = alloca %string*
  %fileName112 = load %string*, %string** %fileName
  %req_URI113 = load %string*, %string** %req_URI
  %cnet_strcat_result114 = call %string* @cnet_strcat(%string* %fileName112,
%string* %req_URI113)
  store %string* %cnet_strcat_result114, %string** %tmp1000111
  %fileName115 = load %string*, %string** %fileName
  %tmp1000116 = load %string*, %string** %tmp1000111
  %cnet_strcpy_result117 = call %string* @cnet_strcpy(%string* %fileName115,
%string* %tmp1000116)
  store %string* %cnet_strcpy_result117, %string** %fileName
  %tmp1000118 = load %string*, %string** %tmp1000111
  %cast119 = bitcast %string* %tmp1000118 to %cnet_file*
  %tmp120 = call i32 @cnet_free(%cnet_file* %cast119)
  br label %if_merge133

elif97:                                           ; preds = %elif90
  br i1 true, label %if_body121, label %if_merge133

if_body121:                                       ; preds = %elif97
  %tmp1000123 = alloca %string*
  %strlit124 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([26 x i8], [26 x i8]* @tmp.21, i32 0, i32 0))
  %method125 = load %string*, %string** %method
  %cnet_strcat_result126 = call %string* @cnet_strcat(%string* %strlit124,
%string* %method125)
  store %string* %cnet_strcat_result126, %string** %tmp1000123
  %cnet_stdout127 = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp1000128 = load %string*, %string** %tmp1000123
  %writeln_result129 = call i32 @writeln(%cnet_file* %cnet_stdout127, %string*
%tmp1000128)
  %tmp1000130 = load %string*, %string** %tmp1000123
  %cast131 = bitcast %string* %tmp1000130 to %cnet_file*
  %tmp132 = call i32 @cnet_free(%cnet_file* %cast131)
  ret i32 -1

if_merge133:                                      ; preds = %if_body89,
%if_body96
  %req_URI134 = load %string*, %string** %req_URI
  %req_URI135 = load %string*, %string** %req_URI
  %slength_result136 = call i32 @slength(%string* %req_URI135)
  %tmp137 = sub i32 %slength_result136, 1
  %charat_result = call i8 @charat(%string* %req_URI134, i32 %tmp137)
  %tmp138 = icmp eq i8 %charat_result, 47
  %tmp_cast139 = sext i1 %tmp138 to i32
  %tmp140 = icmp ne i32 %tmp_cast139, 0
  br i1 %tmp140, label %if_body141, label %elif142

if_body141:                                       ; preds = %if_merge133
  %tmp1000143 = alloca %string*
  %fileName144 = load %string*, %string** %fileName
```

```
  %strlit145 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([11 x i8], [11 x i8]* @tmp.22, i32 0, i32 0))
  %cnet_strcat_result146 = call %string* @cnet_strcat(%string* %fileName144,
%string* %strlit145)
  store %string* %cnet_strcat_result146, %string** %tmp1000143
  %fileName147 = load %string*, %string** %fileName
  %tmp1000148 = load %string*, %string** %tmp1000143
  %cnet_strcpy_result149 = call %string* @cnet_strcpy(%string* %fileName147,
%string* %tmp1000148)
  store %string* %cnet_strcpy_result149, %string** %fileName
  %tmp1000150 = load %string*, %string** %tmp1000143
  %cast151 = bitcast %string* %tmp1000150 to %cnet_file*
  %tmp152 = call i32 @cnet_free(%cnet_file* %cast151)
  br label %if_merge155

elif142:                                          ; preds = %if_merge133
  br i1 true, label %if_body153, label %if_merge155

if_body153:                                       ; preds = %elif142
  br label %if_merge155

if_merge155:                                      ; preds = %if_body141,
%if_body153
  %targetFile = alloca %cnet_file*
  store %cnet_file* null, %cnet_file** %targetFile
  %fileName156 = load %string*, %string** %fileName
  %strlit157 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([3 x i8], [3 x i8]* @tmp.23, i32 0, i32 0))
  %user_fopen_result = call %cnet_file* @user_fopen(%string* %fileName156,
%string* %strlit157)
  store %cnet_file* %user_fopen_result, %cnet_file** %targetFile
  %targetFile158 = load %cnet_file*, %cnet_file** %targetFile
  %error_result = call i32 @error(%cnet_file* %targetFile158)
  %tmp159 = icmp ne i32 %error_result, 0
  br i1 %tmp159, label %if_body160, label %elif161

if_body160:                                       ; preds = %if_merge155
  %respHeader162 = load %string*, %string** %respHeader
  %strlit163 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([14 x i8], [14 x i8]* @tmp.24, i32 0, i32 0))
  %cnet_strcpy_result164 = call %string* @cnet_strcpy(%string* %respHeader162,
%string* %strlit163)
  store %string* %cnet_strcpy_result164, %string** %respHeader
  %tmp1000165 = alloca %string*
  %strlit166 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([16 x i8], [16 x i8]* @tmp.25, i32 0, i32 0))
  %fileName167 = load %string*, %string** %fileName
  %cnet_strcat_result168 = call %string* @cnet_strcat(%string* %strlit166,
%string* %fileName167)
  store %string* %cnet_strcat_result168, %string** %tmp1000165
  %tmp1001169 = alloca %string*
  %tmp1000170 = load %string*, %string** %tmp1000165
  %strlit171 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([20 x i8], [20 x i8]* @tmp.26, i32 0, i32 0))
```

```
  %cnet_strcat_result172 = call %string* @cnet_strcat(%string* %tmp1000170,
%string* %strlit171)
  store %string* %cnet_strcat_result172, %string** %tmp1001169
  %cnet_stdout173 = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp1001174 = load %string*, %string** %tmp1001169
  %writeln_result175 = call i32 @writeln(%cnet_file* %cnet_stdout173, %string*
%tmp1001174)
  %tmp1000176 = load %string*, %string** %tmp1000165
  %cast177 = bitcast %string* %tmp1000176 to %cnet_file*
  %tmp178 = call i32 @cnet_free(%cnet_file* %cast177)
  %tmp1001179 = load %string*, %string** %tmp1001169
  %cast180 = bitcast %string* %tmp1001179 to %cnet_file*
  %tmp181 = call i32 @cnet_free(%cnet_file* %cast180)
  br label %if_merge187

elif161:                                          ; preds = %if_merge155
  br i1 true, label %if_body182, label %if_merge187

if_body182:                                       ; preds = %elif161
  %respHeader184 = load %string*, %string** %respHeader
  %strlit185 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([7 x i8], [7 x i8]* @tmp.27, i32 0, i32 0))
  %cnet_strcpy_result186 = call %string* @cnet_strcpy(%string* %respHeader184,
%string* %strlit185)
  store %string* %cnet_strcpy_result186, %string** %respHeader
  br label %if_merge187

if_merge187:                                      ; preds = %if_body160,
%if_body182
  %cnet_stdout188 = load %cnet_file*, %cnet_file** @cnet_stdout
  %req_line189 = load %string*, %string** %req_line
  %writeln_result190 = call i32 @writeln(%cnet_file* %cnet_stdout188, %string*
%req_line189)
  %tmp1000191 = alloca i32
  %respHeader192 = load %string*, %string** %respHeader
  %strlit193 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([7 x i8], [7 x i8]* @tmp.28, i32 0, i32 0))
  %cnet_strcmp_result194 = call i32 @cnet_strcmp(%string* %respHeader192,
%string* %strlit193)
  store i32 %cnet_strcmp_result194, i32* %tmp1000191
  %tmp1000195 = load i32, i32* %tmp1000191
  %tmp196 = icmp ne i32 %tmp1000195, 0
  br i1 %tmp196, label %if_body197, label %elif198

if_body197:                                       ; preds = %if_merge187
  %tmp1000199 = alloca %string*
  %targetFile200 = load %cnet_file*, %cnet_file** %targetFile
  %readall_result = call %string* @readall(%cnet_file* %targetFile200)
  store %string* %readall_result, %string** %tmp1000199
  %tmp1001201 = alloca %string*
  %strlit202 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([18 x i8], [18 x i8]* @tmp.29, i32 0, i32 0))
  %tmp1000203 = load %string*, %string** %tmp1000199
  %cnet_strcat_result204 = call %string* @cnet_strcat(%string* %strlit202,
```

```
%string* %tmp1000203)
  store %string* %cnet_strcat_result204, %string** %tmp1001201
  %clntsock205 = load %cnet_file*, %cnet_file** %clntsock
  %tmp1001206 = load %string*, %string** %tmp1001201
  %writeln_result207 = call i32 @writeln(%cnet_file* %clntsock205, %string*
%tmp1001206)
  %tmp1000208 = load %string*, %string** %tmp1000199
  %cast209 = bitcast %string* %tmp1000208 to %cnet_file*
  %tmp210 = call i32 @cnet_free(%cnet_file* %cast209)
  %tmp1001211 = load %string*, %string** %tmp1001201
  %cast212 = bitcast %string* %tmp1001211 to %cnet_file*
  %tmp213 = call i32 @cnet_free(%cnet_file* %cast212)
  br label %if_merge219

elif198:                                          ; preds = %if_merge187
  br i1 true, label %if_body214, label %if_merge219

if_body214:                                       ; preds = %elif198
  %clntsock216 = load %cnet_file*, %cnet_file** %clntsock
  %strlit217 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([25 x i8], [25 x i8]* @tmp.30, i32 0, i32 0))
  %writeln_result218 = call i32 @writeln(%cnet_file* %clntsock216, %string*
%strlit217)
  br label %if_merge219

if_merge219:                                      ; preds = %if_body197,
%if_body214
  %cnet_stdout220 = load %cnet_file*, %cnet_file** @cnet_stdout
  %strlit221 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([29 x i8], [29 x i8]* @tmp.31, i32 0, i32 0))
  %writeln_result222 = call i32 @writeln(%cnet_file* %cnet_stdout220, %string*
%strlit221)
  %clntsock223 = load %cnet_file*, %cnet_file** %clntsock
  %tmp224 = call i32 @cnet_free(%cnet_file* %clntsock223)
  %req_line225 = load %string*, %string** %req_line
  %cast226 = bitcast %string* %req_line225 to %cnet_file*
  %tmp227 = call i32 @cnet_free(%cnet_file* %cast226)
  %method228 = load %string*, %string** %method
  %cast229 = bitcast %string* %method228 to %cnet_file*
  %tmp230 = call i32 @cnet_free(%cnet_file* %cast229)
  %req_URI231 = load %string*, %string** %req_URI
  %cast232 = bitcast %string* %req_URI231 to %cnet_file*
  %tmp233 = call i32 @cnet_free(%cnet_file* %cast232)
  %httpVersion234 = load %string*, %string** %httpVersion
  %cast235 = bitcast %string* %httpVersion234 to %cnet_file*
  %tmp236 = call i32 @cnet_free(%cnet_file* %cast235)
  %fileName237 = load %string*, %string** %fileName
  %cast238 = bitcast %string* %fileName237 to %cnet_file*
  %tmp239 = call i32 @cnet_free(%cnet_file* %cast238)
  %respHeader240 = load %string*, %string** %respHeader
  %cast241 = bitcast %string* %respHeader240 to %cnet_file*
  %tmp242 = call i32 @cnet_free(%cnet_file* %cast241)
  br label %while
```

```
merge:                                            ; preds = %while
  %port243 = load %string*, %string** %port
  %cast244 = bitcast %string* %port243 to %cnet_file*
  %tmp245 = call i32 @cnet_free(%cnet_file* %cast244)
  %webroot246 = load %string*, %string** %webroot
  %cast247 = bitcast %string* %webroot246 to %cnet_file*
  %tmp248 = call i32 @cnet_free(%cnet_file* %cast247)
  ret i32 0
}

declare %string** @cnet_index_arr(%array*, i32)

declare %array* @cnet_init_array(i32, i32, i32, i32, %string*, ...)
```

Example Code 2: client-server.cnet

```
struct conn{
      string serv_addr;
      int port;
      string req_fname;
      file f;
      socket sock;
};

void cleanup(struct conn client_conn)
{
      delete client_conn.sock;
      delete client_conn.f;
      delete client_conn;

      return;
}

void print_conn_details(struct conn s_conn)
{
      stdout.writeln("Connecting to host: " + s_conn.serv_addr
                              + "\nPort " + soi(s_conn.port));
      stdout.writeln("Requesting " + s_conn.req_fname);
      stdout.writeln("Protocol : HTTP/1.0");

      return;
}

string req_line(string host, string fname){
      return "GET " + fname + " HTTP/1.0\r\n"
                        + "Host: " + host + "\r\n\r\n";
}

int main(string [] args)
```

```
{

      if (args.alength() != 5){
        stdout.writeln("usage: " + args[0] + " <server-addr> <server-port>
<req-filename> <output-filename>");
        return -1;
    }

      struct conn s_conn = new struct conn;

      /* Parsing connection details */
      s_conn.serv_addr = args[1];
    s_conn.port = args[2].toint();
      s_conn.req_fname = args[3];
      print_conn_details(s_conn);

      /* Making request to host server */
      s_conn.sock = nopen(s_conn.serv_addr, s_conn.port, "tcp", "connect");
      s_conn.sock.writeln(req_line(s_conn.serv_addr, s_conn.req_fname));

      /* Reading from conn socket and writing to local file*/
      s_conn.f = fopen(args[4], "wb");
      s_conn.f.writeln(s_conn.sock.readall());

      cleanup(s_conn);

      return 0;
}
```

LLVM Output 2: client-server.cnet

```
; ModuleID = 'CNet'
source_filename = "CNet"

%cnet_file = type { %string*, %string*, i32 }
%string = type { %string*, %string*, i32 }
%array = type { %string*, %string*, i32, i32 }
%conn = type { %string*, i32, %string*, %cnet_file*, %cnet_file* }

@cnet_stdin = external externally_initialized global %cnet_file*
@cnet_stdout = external externally_initialized global %cnet_file*
@tmp = private unnamed_addr constant [8 x i8] c"usage: \00", align 1
@tmp.1 = private unnamed_addr constant [62 x i8] c" <server-addr>
<server-port> <req-filename> <output-filename>\00", align 1
@tmp.2 = private unnamed_addr constant [4 x i8] c"tcp\00", align 1
@tmp.3 = private unnamed_addr constant [8 x i8] c"connect\00", align 1
@tmp.4 = private unnamed_addr constant [3 x i8] c"wb\00", align 1
@tmp.5 = private unnamed_addr constant [1 x i8] zeroinitializer, align 1
@tmp.6 = private unnamed_addr constant [5 x i8] c"GET \00", align 1
```

```
@tmp.7 = private unnamed_addr constant [12 x i8] c" HTTP/1.0\0D\0A\00", align
1
@tmp.8 = private unnamed_addr constant [7 x i8] c"Host: \00", align 1
@tmp.9 = private unnamed_addr constant [5 x i8] c"\0D\0A\0D\0A\00", align 1
@tmp.10 = private unnamed_addr constant [21 x i8] c"Connecting to host: \00",
align 1
@tmp.11 = private unnamed_addr constant [7 x i8] c"\0APort \00", align 1
@tmp.12 = private unnamed_addr constant [12 x i8] c"Requesting \00", align 1
@tmp.13 = private unnamed_addr constant [20 x i8] c"Protocol : HTTP/1.0\00",
align 1

declare i32 @cnet_free(%cnet_file*)

declare i32 @alength(%array*)

declare i8 @charat(%string*, i32)

declare void @split(%string*, %string*, %array*)

declare i8 @find_char(%string*, i8)

declare %string* @reverse(%string*)

declare %string* @substring(%string*, i32, i32)

declare %string* @lower(%string*)

declare %string* @upper(%string*)

declare %string* @user_soi(i32)

declare i32 @toint(%string*)

declare double @tofloat(%string*)

declare i32 @slength(%string*)

declare i32 @cnet_strcmp(%string*, %string*)

declare %string* @cnet_strcat(%string*, %string*)

declare %string* @cnet_strmult(%string*, i32)

declare %string* @cnet_strcpy(%string*, %string*)

declare i32 @error(%cnet_file*)

declare %string* @readall(%cnet_file*)

declare %string* @readln(%cnet_file*)

declare i32 @nwrite(%cnet_file*, %string*, i32)

declare i32 @writeln(%cnet_file*, %string*)
```

```
declare %cnet_file* @user_fopen(%string*, %string*)

declare %string* @read(%cnet_file*, i32)

declare i32 @write(%cnet_file*, %string*)

declare %cnet_file* @naccept(%cnet_file*)

declare %cnet_file* @user_nopen(%string*, i32, %string*, %string*)

declare %string* @cnet_new_str_nolen(i8*)

declare %string* @cnet_empty_str()

declare i8 @cnet_char_at(%string*, i32)

declare i8* @memset(i8*, i32, i64)

declare i32 @main(i32, i8**)

define i32 @user_main(%array* %args) {
entry:
  %args1 = alloca %array*
  store %array* %args, %array** %args1
  %args2 = load %array*, %array** %args1
  %alength_result = call i32 @alength(%array* %args2)
  %tmp = icmp ne i32 %alength_result, 5
  %tmp_cast = sext i1 %tmp to i32
  %tmp3 = icmp ne i32 %tmp_cast, 0
  br i1 %tmp3, label %if_body, label %elif

if_body:                                          ; preds = %entry
  %tmp1000 = alloca %string*
  %strlit = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([8 x
i8], [8 x i8]* @tmp, i32 0, i32 0))
  %arr = load %array*, %array** %args1
  %idx_elt = call %string** @cnet_index_arr(%array* %arr, i32 0)
  %args4 = load %string*, %string** %idx_elt
  %cnet_strcat_result = call %string* @cnet_strcat(%string* %strlit, %string*
%args4)
  store %string* %cnet_strcat_result, %string** %tmp1000
  %tmp1001 = alloca %string*
  %tmp10005 = load %string*, %string** %tmp1000
  %strlit6 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([62
x i8], [62 x i8]* @tmp.1, i32 0, i32 0))
  %cnet_strcat_result7 = call %string* @cnet_strcat(%string* %tmp10005,
%string* %strlit6)
  store %string* %cnet_strcat_result7, %string** %tmp1001
  %cnet_stdout = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp10018 = load %string*, %string** %tmp1001
  %writeln_result = call i32 @writeln(%cnet_file* %cnet_stdout, %string*
%tmp10018)
  %tmp10009 = load %string*, %string** %tmp1000
```

```
  %cast = bitcast %string* %tmp10009 to %cnet_file*
  %tmp10 = call i32 @cnet_free(%cnet_file* %cast)
  %tmp100111 = load %string*, %string** %tmp1001
  %cast12 = bitcast %string* %tmp100111 to %cnet_file*
  %tmp13 = call i32 @cnet_free(%cnet_file* %cast12)
  ret i32 -1

elif:                                             ; preds = %entry
  br i1 true, label %if_body14, label %if_merge

if_body14:                                        ; preds = %elif
  br label %if_merge

if_merge:                                         ; preds = %if_body14
  %s_conn = alloca %conn*
  store %conn* null, %conn** %s_conn
  %malloccall = tail call i8* @malloc(i32 ptrtoint (%conn* getelementptr
(%conn, %conn* null, i32 1) to i32))
  %tmp16 = bitcast i8* %malloccall to %conn*
  %tmp17 = getelementptr inbounds %conn, %conn* %tmp16, i32 0, i32 0
  %empty_str = call %string* @cnet_empty_str()
  store %string* %empty_str, %string** %tmp17
  %tmp18 = getelementptr inbounds %conn, %conn* %tmp16, i32 0, i32 2
  %empty_str19 = call %string* @cnet_empty_str()
  store %string* %empty_str19, %string** %tmp18
  store %conn* %tmp16, %conn** %s_conn
  %tmp20 = load %conn*, %conn** %s_conn
  %0 = getelementptr inbounds %conn, %conn* %tmp20, i32 0, i32 0
  %serv_addr = load %string*, %string** %0
  %arr21 = load %array*, %array** %args1
  %idx_elt22 = call %string** @cnet_index_arr(%array* %arr21, i32 1)
  %args23 = load %string*, %string** %idx_elt22
  %cnet_strcpy_result = call %string* @cnet_strcpy(%string* %serv_addr,
%string* %args23)
  %tmp24 = load %conn*, %conn** %s_conn
  %1 = getelementptr inbounds %conn, %conn* %tmp24, i32 0, i32 0
  store %string* %cnet_strcpy_result, %string** %1
  %arr25 = load %array*, %array** %args1
  %idx_elt26 = call %string** @cnet_index_arr(%array* %arr25, i32 2)
  %args27 = load %string*, %string** %idx_elt26
  %toint_result = call i32 @toint(%string* %args27)
  %tmp28 = load %conn*, %conn** %s_conn
  %2 = getelementptr inbounds %conn, %conn* %tmp28, i32 0, i32 1
  store i32 %toint_result, i32* %2
  %tmp29 = load %conn*, %conn** %s_conn
  %3 = getelementptr inbounds %conn, %conn* %tmp29, i32 0, i32 2
  %req_fname = load %string*, %string** %3
  %arr30 = load %array*, %array** %args1
  %idx_elt31 = call %string** @cnet_index_arr(%array* %arr30, i32 3)
  %args32 = load %string*, %string** %idx_elt31
  %cnet_strcpy_result33 = call %string* @cnet_strcpy(%string* %req_fname,
%string* %args32)
  %tmp34 = load %conn*, %conn** %s_conn
  %4 = getelementptr inbounds %conn, %conn* %tmp34, i32 0, i32 2
```

```
  store %string* %cnet_strcpy_result33, %string** %4
  %s_conn35 = load %conn*, %conn** %s_conn
  call void @user_print_conn_details(%conn* %s_conn35)
  %tmp36 = load %conn*, %conn** %s_conn
  %5 = getelementptr inbounds %conn, %conn* %tmp36, i32 0, i32 0
  %serv_addr37 = load %string*, %string** %5
  %tmp38 = load %conn*, %conn** %s_conn
  %6 = getelementptr inbounds %conn, %conn* %tmp38, i32 0, i32 1
  %port = load i32, i32* %6
  %strlit39 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([4
x i8], [4 x i8]* @tmp.2, i32 0, i32 0))
  %strlit40 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([8
x i8], [8 x i8]* @tmp.3, i32 0, i32 0))
  %user_nopen_result = call %cnet_file* @user_nopen(%string* %serv_addr37, i32
%port, %string* %strlit39, %string* %strlit40)
  %tmp41 = load %conn*, %conn** %s_conn
  %7 = getelementptr inbounds %conn, %conn* %tmp41, i32 0, i32 4
  store %cnet_file* %user_nopen_result, %cnet_file** %7
  %tmp100042 = alloca %string*
  %tmp43 = load %conn*, %conn** %s_conn
  %8 = getelementptr inbounds %conn, %conn* %tmp43, i32 0, i32 0
  %serv_addr44 = load %string*, %string** %8
  %tmp45 = load %conn*, %conn** %s_conn
  %9 = getelementptr inbounds %conn, %conn* %tmp45, i32 0, i32 2
  %req_fname46 = load %string*, %string** %9
  %user_req_line_result = call %string* @user_req_line(%string* %serv_addr44,
%string* %req_fname46)
  store %string* %user_req_line_result, %string** %tmp100042
  %tmp47 = load %conn*, %conn** %s_conn
  %10 = getelementptr inbounds %conn, %conn* %tmp47, i32 0, i32 4
  %sock = load %cnet_file*, %cnet_file** %10
  %tmp100048 = load %string*, %string** %tmp100042
  %writeln_result49 = call i32 @writeln(%cnet_file* %sock, %string*
%tmp100048)
  %tmp100050 = load %string*, %string** %tmp100042
  %cast51 = bitcast %string* %tmp100050 to %cnet_file*
  %tmp52 = call i32 @cnet_free(%cnet_file* %cast51)
  %arr53 = load %array*, %array** %args1
  %idx_elt54 = call %string** @cnet_index_arr(%array* %arr53, i32 4)
  %args55 = load %string*, %string** %idx_elt54
  %strlit56 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([3
x i8], [3 x i8]* @tmp.4, i32 0, i32 0))
  %user_fopen_result = call %cnet_file* @user_fopen(%string* %args55, %string*
%strlit56)
  %tmp57 = load %conn*, %conn** %s_conn
  %11 = getelementptr inbounds %conn, %conn* %tmp57, i32 0, i32 3
  store %cnet_file* %user_fopen_result, %cnet_file** %11
  %tmp100058 = alloca %string*
  %tmp59 = load %conn*, %conn** %s_conn
  %12 = getelementptr inbounds %conn, %conn* %tmp59, i32 0, i32 4
  %sock60 = load %cnet_file*, %cnet_file** %12
  %readall_result = call %string* @readall(%cnet_file* %sock60)
  store %string* %readall_result, %string** %tmp100058
  %tmp61 = load %conn*, %conn** %s_conn
```

```
  %13 = getelementptr inbounds %conn, %conn* %tmp61, i32 0, i32 3
  %f = load %cnet_file*, %cnet_file** %13
  %tmp100062 = load %string*, %string** %tmp100058
  %writeln_result63 = call i32 @writeln(%cnet_file* %f, %string* %tmp100062)
  %tmp100064 = load %string*, %string** %tmp100058
  %cast65 = bitcast %string* %tmp100064 to %cnet_file*
  %tmp66 = call i32 @cnet_free(%cnet_file* %cast65)
  %s_conn67 = load %conn*, %conn** %s_conn
  call void @user_cleanup(%conn* %s_conn67)
  ret i32 0
}

define %string* @user_req_line(%string* %host, %string* %fname) {
entry:
  %host1 = alloca %string*
  store %string* %host, %string** %host1
  %fname2 = alloca %string*
  store %string* %fname, %string** %fname2
  %ret_tmp = alloca %string*
  %strlit = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([1 x
i8], [1 x i8]* @tmp.5, i32 0, i32 0))
  store %string* %strlit, %string** %ret_tmp
  %tmp1000 = alloca %string*
  %strlit3 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([5
x i8], [5 x i8]* @tmp.6, i32 0, i32 0))
  %fname4 = load %string*, %string** %fname2
  %cnet_strcat_result = call %string* @cnet_strcat(%string* %strlit3, %string*
%fname4)
  store %string* %cnet_strcat_result, %string** %tmp1000
  %tmp1001 = alloca %string*
  %tmp10005 = load %string*, %string** %tmp1000
  %strlit6 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([12
x i8], [12 x i8]* @tmp.7, i32 0, i32 0))
  %cnet_strcat_result7 = call %string* @cnet_strcat(%string* %tmp10005,
%string* %strlit6)
  store %string* %cnet_strcat_result7, %string** %tmp1001
  %tmp1002 = alloca %string*
  %tmp10018 = load %string*, %string** %tmp1001
  %strlit9 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([7
x i8], [7 x i8]* @tmp.8, i32 0, i32 0))
  %cnet_strcat_result10 = call %string* @cnet_strcat(%string* %tmp10018,
%string* %strlit9)
  store %string* %cnet_strcat_result10, %string** %tmp1002
  %tmp1003 = alloca %string*
  %tmp100211 = load %string*, %string** %tmp1002
  %host12 = load %string*, %string** %host1
  %cnet_strcat_result13 = call %string* @cnet_strcat(%string* %tmp100211,
%string* %host12)
  store %string* %cnet_strcat_result13, %string** %tmp1003
  %tmp1004 = alloca %string*
  %tmp100314 = load %string*, %string** %tmp1003
  %strlit15 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([5
x i8], [5 x i8]* @tmp.9, i32 0, i32 0))
  %cnet_strcat_result16 = call %string* @cnet_strcat(%string* %tmp100314,
```

```
%string* %strlit15)
  store %string* %cnet_strcat_result16, %string** %tmp1004
  %ret_tmp17 = load %string*, %string** %ret_tmp
  %tmp100418 = load %string*, %string** %tmp1004
  %cnet_strcpy_result = call %string* @cnet_strcpy(%string* %ret_tmp17,
%string* %tmp100418)
  store %string* %cnet_strcpy_result, %string** %ret_tmp
  %tmp100019 = load %string*, %string** %tmp1000
  %cast = bitcast %string* %tmp100019 to %cnet_file*
  %tmp = call i32 @cnet_free(%cnet_file* %cast)
  %tmp100120 = load %string*, %string** %tmp1001
  %cast21 = bitcast %string* %tmp100120 to %cnet_file*
  %tmp22 = call i32 @cnet_free(%cnet_file* %cast21)
  %tmp100223 = load %string*, %string** %tmp1002
  %cast24 = bitcast %string* %tmp100223 to %cnet_file*
  %tmp25 = call i32 @cnet_free(%cnet_file* %cast24)
  %tmp100326 = load %string*, %string** %tmp1003
  %cast27 = bitcast %string* %tmp100326 to %cnet_file*
  %tmp28 = call i32 @cnet_free(%cnet_file* %cast27)
  %tmp100429 = load %string*, %string** %tmp1004
  %cast30 = bitcast %string* %tmp100429 to %cnet_file*
  %tmp31 = call i32 @cnet_free(%cnet_file* %cast30)
  %ret_tmp32 = load %string*, %string** %ret_tmp
  ret %string* %ret_tmp32
}

define void @user_print_conn_details(%conn* %s_conn) {
entry:
  %s_conn1 = alloca %conn*
  store %conn* %s_conn, %conn** %s_conn1
  %tmp1000 = alloca %string*
  %strlit = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([21
x i8], [21 x i8]* @tmp.10, i32 0, i32 0))
  %tmp = load %conn*, %conn** %s_conn1
  %0 = getelementptr inbounds %conn, %conn* %tmp, i32 0, i32 0
  %serv_addr = load %string*, %string** %0
  %cnet_strcat_result = call %string* @cnet_strcat(%string* %strlit, %string*
%serv_addr)
  store %string* %cnet_strcat_result, %string** %tmp1000
  %tmp1001 = alloca %string*
  %tmp10002 = load %string*, %string** %tmp1000
  %strlit3 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds ([7
x i8], [7 x i8]* @tmp.11, i32 0, i32 0))
  %cnet_strcat_result4 = call %string* @cnet_strcat(%string* %tmp10002,
%string* %strlit3)
  store %string* %cnet_strcat_result4, %string** %tmp1001
  %tmp1002 = alloca %string*
  %tmp5 = load %conn*, %conn** %s_conn1
  %1 = getelementptr inbounds %conn, %conn* %tmp5, i32 0, i32 1
  %port = load i32, i32* %1
  %user_soi_result = call %string* @user_soi(i32 %port)
  store %string* %user_soi_result, %string** %tmp1002
  %tmp1003 = alloca %string*
  %tmp10016 = load %string*, %string** %tmp1001
```

```
  %tmp10027 = load %string*, %string** %tmp1002
  %cnet_strcat_result8 = call %string* @cnet_strcat(%string* %tmp10016,
%string* %tmp10027)
  store %string* %cnet_strcat_result8, %string** %tmp1003
  %cnet_stdout = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp10039 = load %string*, %string** %tmp1003
  %writeln_result = call i32 @writeln(%cnet_file* %cnet_stdout, %string*
%tmp10039)
  %tmp100010 = load %string*, %string** %tmp1000
  %cast = bitcast %string* %tmp100010 to %cnet_file*
  %tmp11 = call i32 @cnet_free(%cnet_file* %cast)
  %tmp100112 = load %string*, %string** %tmp1001
  %cast13 = bitcast %string* %tmp100112 to %cnet_file*
  %tmp14 = call i32 @cnet_free(%cnet_file* %cast13)
  %tmp100215 = load %string*, %string** %tmp1002
  %cast16 = bitcast %string* %tmp100215 to %cnet_file*
  %tmp17 = call i32 @cnet_free(%cnet_file* %cast16)
  %tmp100318 = load %string*, %string** %tmp1003
  %cast19 = bitcast %string* %tmp100318 to %cnet_file*
  %tmp20 = call i32 @cnet_free(%cnet_file* %cast19)
  %tmp100021 = alloca %string*
  %strlit22 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([12 x i8], [12 x i8]* @tmp.12, i32 0, i32 0))
  %tmp23 = load %conn*, %conn** %s_conn1
  %2 = getelementptr inbounds %conn, %conn* %tmp23, i32 0, i32 2
  %req_fname = load %string*, %string** %2
  %cnet_strcat_result24 = call %string* @cnet_strcat(%string* %strlit22,
%string* %req_fname)
  store %string* %cnet_strcat_result24, %string** %tmp100021
  %cnet_stdout25 = load %cnet_file*, %cnet_file** @cnet_stdout
  %tmp100026 = load %string*, %string** %tmp100021
  %writeln_result27 = call i32 @writeln(%cnet_file* %cnet_stdout25, %string*
%tmp100026)
  %tmp100028 = load %string*, %string** %tmp100021
  %cast29 = bitcast %string* %tmp100028 to %cnet_file*
  %tmp30 = call i32 @cnet_free(%cnet_file* %cast29)
  %cnet_stdout31 = load %cnet_file*, %cnet_file** @cnet_stdout
  %strlit32 = call %string* @cnet_new_str_nolen(i8* getelementptr inbounds
([20 x i8], [20 x i8]* @tmp.13, i32 0, i32 0))
  %writeln_result33 = call i32 @writeln(%cnet_file* %cnet_stdout31, %string*
%strlit32)
  ret void
}

define void @user_cleanup(%conn* %client_conn) {
entry:
  %client_conn1 = alloca %conn*
  store %conn* %client_conn, %conn** %client_conn1
  %tmp = load %conn*, %conn** %client_conn1
  %0 = getelementptr inbounds %conn, %conn* %tmp, i32 0, i32 4
  %sock = load %cnet_file*, %cnet_file** %0
  %tmp2 = call i32 @cnet_free(%cnet_file* %sock)
  %tmp3 = load %conn*, %conn** %client_conn1
  %1 = getelementptr inbounds %conn, %conn* %tmp3, i32 0, i32 3
```

```
  %f = load %cnet_file*, %cnet_file** %1
  %tmp4 = call i32 @cnet_free(%cnet_file* %f)
  %client_conn5 = load %conn*, %conn** %client_conn1
  %tmp6 = getelementptr inbounds %conn, %conn* %client_conn5, i32 0, i32 0
  %tmp7 = load %string*, %string** %tmp6
  %cast = bitcast %string* %tmp7 to %cnet_file*
  %tmp8 = call i32 @cnet_free(%cnet_file* %cast)
  %tmp9 = getelementptr inbounds %conn, %conn* %client_conn5, i32 0, i32 2
  %tmp10 = load %string*, %string** %tmp9
  %cast11 = bitcast %string* %tmp10 to %cnet_file*
  %tmp12 = call i32 @cnet_free(%cnet_file* %cast11)
  %2 = bitcast %conn* %client_conn5 to i8*
  tail call void @free(i8* %2)
  ret void
}

declare %string** @cnet_index_arr(%array*, i32)

declare noalias i8* @malloc(i32)

declare void @free(i8*)
```

10. Appendix

Project log

```
commit 984e34f2063898ab00f54837316e486c96e1d594
Merge: ae9e3bf c32c1c3
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Apr 26 21:40:15 2021 -0400

    final http-demo.cnet #165 from Bruk3/bruk-demo-server

commit c32c1c3133f9f045dc2d075ddff19b8a1778c1b2
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 26 21:31:29 2021 -0400

    final http-demo.cnet

commit ae9e3bfdadf1be5dd60195d290134411ad76238c
Merge: 66ad202 527be22
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Mon Apr 26 13:57:37 2021 -0400

    Merge pull request #164 from Bruk3/fix-arrr-decl

    Fix arrr decl

commit 527be22af2f2a9baec6a7d383a5546d60794c1df
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 26 10:39:08 2021 -0400

    fixed array decl bug

commit 1f5826d38f2341c7dcdc2a3071e08f41fd165bee
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 26 09:04:56 2021 -0400
```

```
    readall fixed

commit fae69f08fc4e4e33db69d8b0eb9514a54642c890
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 26 08:11:13 2021 -0400

    client_server

commit 06098e6fb0f453dfe124fffa08c82ab380775d49
Merge: dbfcb7d 0585348
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 26 06:51:49 2021 -0400

    Merge branch 'bruk-demo-server' of https://github.com/Bruk3/C-net into
bruk-demo-server

commit dbfcb7d5f841888fa8f67fa04ee6597704324e9f
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 26 06:51:44 2021 -0400

    stash

commit 05853486b1e973f055032dbc0b7398d24a93e400
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 26 00:44:31 2021 -0400

    added neater version of simple HTTP server written in cnet

commit f2257a61ca2c3882ef2a06def68212b6b61936b7
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 26 00:39:27 2021 -0400

    got the server working
```

```
commit 21f86f4ee81b2ed40058072c671267620b6b5b64
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 25 21:47:00 2021 -0400

    http-server demo 1/3rd of the way there

commit 66ad2022fdf3109ef624774a806a57bf1fd9f96f
Merge: 73761db a4baba4
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 17:36:43 2021 -0400

    Integration tests for fileIo and string tests for string library #162

commit a4baba4ed237452b642db2c90fb2237b0664b59e
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 25 17:32:00 2021 -0400

    fileIo tests

commit 73761dbf5c198749c2752dab39e3e6a640a289d6
Merge: 3699da1 b4d4e9d
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 13:41:39 2021 -0400

    File IO integration  #159 from Bruk3/bruk-fileIo-Integration

commit b4d4e9d6a8aa3437cd30da6f74e9297278ba3c20
Merge: 9bf3e7e 3699da1
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 25 13:39:10 2021 -0400
```

```
    Merge branch 'main' into bruk-fileIo-Integration

commit 3699da1fd6198029990345100970a917ae88f7a3
Author: Rediet24 <38728691+Rediet24@users.noreply.github.com>
Date:   Sun Apr 25 13:38:34 2021 -0400

    Add more integration tests (#158)

    * changed output file name

    * added tests for itos function

    * Merge with main to update soi function

    * Added integration tests for soi and if statement

    * fixed while loop oeprand in codegen

    * added more integration tests

commit 9bf3e7e79772b2c3db82c0f3b92a0ed372177464
Merge: f7a9225 29f3ead
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 25 13:37:02 2021 -0400

    read from file and writeln to stdout works

commit f7a9225e1da9d5aa79c0d7d73593c17b44efa660
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 25 13:19:50 2021 -0400

    read and writeln to stdout works
```

```
commit 29f3eadc05187eabd864603df5ecaf182d681cd5
Merge: 936cb63 106c494
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Sun Apr 25 12:40:12 2021 -0400

    Merge pull request #157 from Bruk3/test-all-string-func

    Test all string func

commit 106c49407e9c2dcdcd468cf218915eeff7bef8e5
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 12:36:27 2021 -0400

    changed stringof to soi

commit e5361f900b3002aa1397a165fd05d5bcf07feefa
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 12:13:07 2021 -0400

    minor changes

commit 936cb6318d64e900b2cb8a587e462e3fbb263381
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 12:06:35 2021 -0400

    create new libcnet-main.a library to prevent linking error in stdlib tests
(#156)

commit 216aacc5072ed0e91b973f548e5444d2967be374
Merge: 4572bfa 3f7d82a
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 11:38:34 2021 -0400
```

```
    Merge branch 'main' of https://github.com/Bruk3/C-net into
test-all-string-func

commit 4572bfa2f930f5a24d4b94ebf2826f45c2f8efbf
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 11:38:21 2021 -0400

    stdlib string function names updated

commit 3f7d82a335e39fc632a0b75812faec3a34271218
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 10:02:18 2021 -0400

    renaming to /dev/stdout is not allowed in macOS (#155)

commit 7e47e7df1700eb1f40a6f738a9db3677291ad559
Merge: bfa303c ea5fe5c
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Sun Apr 25 09:43:07 2021 -0400

    Merge pull request #154 from Bruk3/parse-cmd-args

    Parse cmd args

commit ea5fe5cee5424ec958625263c254a5f78c82e42f
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 09:39:30 2021 -0400

    parse cmd line args for both main prototypes

commit bfa303cc1e6899f7f45a132bd533a4a4b0ffe0ba
Merge: 480f1ba c173cb6
```

```
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sun Apr 25 08:58:17 2021 -0400

    Merge pull request #153 from Bruk3/integration-test-enable

    readded integration test automation

commit c173cb6da03e6141b40bbb23938651edbf6bdeda
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 25 08:53:02 2021 -0400

    readded integration test automation

commit e9e2998a093f1ef8626005a36a54717c6049747c
Merge: b7fb661 480f1ba
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 08:46:02 2021 -0400

    Merge branch 'main' of https://github.com/Bruk3/C-net into parse-cmd-args

commit b7fb661da2f08be8748c0810e49deca2358e8794
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Apr 25 08:45:55 2021 -0400

    parse main cmd-line args

commit 480f1ba8e501f5ed53aa676648b61c8200e35bb4
Merge: d003abc bd0707a
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sun Apr 25 08:31:46 2021 -0400

    Merge pull request #152 from Bruk3/test-ios
```

```
    IO modifications for some enhancements and a tentative fix for overloaded
functions

commit bd0707a4ccae71abbf653abc4147316f7a52e6de
Merge: 7b21375 d003abc
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sun Apr 25 08:28:19 2021 -0400

    Merge branch 'main' into test-ios

commit 7b2137551d57c629319fbeab66d5e698e6a51066
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 25 08:25:01 2021 -0400

    added some null checks in socket/file operations

commit 297c329513aa5b096b18b35aa94553c562a7ca58
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 25 08:12:49 2021 -0400

    removed unecessary else block in codegen

commit d003abc5e68132c735b656f23953f7ede356407c
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 08:09:36 2021 -0400

    readline and split string implementation (#151)

    * implement split string

    * tested split string
```

```
    * fix str_split prototype

    * small leak fix

    * minor fix on test8

    * start http-server.c

    * mvp for http-server

    * minor protoype/definition mismatch fix

    * fix some parser and semant bugs in http-server

    * test readln and test9. nread needs to be fixed.

commit b0a67ccff9a89750b0d56a86297e67ef58b205f6
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 25 08:06:45 2021 -0400

    added some more prototypes to built-in functions

commit 6f74d05520704401e6346f430b6ad35d9cb3b215
Merge: 7681467 66cc2fd
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Sun Apr 25 06:52:34 2021 -0400

    Merge pull request #148 from Bruk3/string-arrays

    string arrays and indexing impl

commit 768146759cc99e06d0ff0bfcfd4ffd0d444ef3f2
```

```
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 25 01:48:11 2021 -0400

    Implement itos: integer to string to print integers (#150)

commit 66cc2fde2890a9fc65f1248b2dff5d378a95f9d0
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sat Apr 24 22:52:51 2021 -0400

    string arrays and indexing impl

commit af068ef11d8205ce69e4d7d585cda47643c5b537
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Apr 24 22:40:23 2021 -0400

    String handling fix and expansion (#143)

    * added string multiplication and addition

    * generalized string handler to return a tuple of pre-stmts, the
expression, and post-stmts

    * fixed string bug for if statements

    * fixed string bug for for and while loops

    * added string cleanup for variable declaration/assignments

    * fixed bug of unintialized variables being used in codegen

    * adjusted output of failing tests

commit 95464f84e91d35b5f92a97e081ade08cbb57b6ed
```

```
Merge: a13c87d 962ab42
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Apr 24 22:08:50 2021 -0400

    Merge pull request #144 from Bruk3/ccnet-path-fix

    fixed path error in toplevel

commit 962ab42fe11e3c475208dc61d6e34bb08925a537
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Apr 24 22:04:36 2021 -0400

    fixed path error in toplevel

commit a13c87d8cec40b15056857d5d65c2f15b6a96dc8
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Apr 24 13:27:00 2021 -0400

    added if/else if/else statements without body (#138)

    * added if/else if/else statements without body

    * added statement checking for if/else statements

commit c53b572020416778ee02c22b8cba69fe254d2123
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Apr 24 12:28:37 2021 -0400

    fix user_ prefix mismatch between fdecl and fcall (#139)

commit 097ebad0c588a4e2d89675d4f04826f387bb7ab9
Merge: 9b6d76a 4bd887a
```

```
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Apr 22 11:57:39 2021 -0400

    Merge pull request #137 from Bruk3/while-fix

    fix for while statement branching

commit 4bd887af279d97ead1706a8d9a31a57fd6bae565
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Apr 22 11:14:10 2021 -0400

    fix for while statement branching

commit 9b6d76a635db1bd94a4b51d07b4017ce2d492eaf
Merge: b5b10f0 11af2d7
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Apr 22 11:06:00 2021 -0400

    Merge pull request #136 from Bruk3/scanner-minor-enhancement

    improved parsing escape sequences in string literals

commit 11af2d7029db6a6ace3f82e56f1c0a5a4a54f612
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Apr 22 11:01:19 2021 -0400

    improved parsing escape sequences in string literals

commit b5b10f0395ddbe0f8a27963c062fd229462ca292
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Apr 22 08:41:42 2021 -0400
```

```
    indexing addition (#134)

    * incomplete work on strings

    * removed expression index from sast

    * fixed struct member error for string printing

    * added array indexing calls

    * fixed lookup conflict

commit d4e9fa64edf9d6e69f2a0265c5d23c45aacb9cba
Merge: cada45f 8e3ca0b
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Thu Apr 22 00:24:41 2021 -0400

    Merge pull request #133 from Bruk3/tmp-will-branch

    Merge will branch

commit 8e3ca0b4eeeb2dfe2acaf6c7f2a2c1a65492e452
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 23:36:18 2021 -0400

    merge conflict fix

commit 2ae1375d784e622763715df566cc23ae85c25c46
Merge: 25a3584 cada45f
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 22:56:48 2021 -0400
```

```
    Merge branch 'main' of https://github.com/Bruk3/C-net into will-codegen-new

commit cada45f8e009126b999191b2522b601284988ac6
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Apr 21 22:35:30 2021 -0400

    Minor clean up for repo  #132

commit db482a03409826f9c57a7ab1a0d2c7108babe41a
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Wed Apr 21 14:38:45 2021 -0400

    Index fix (#130)

    * added a more flexible python toplevel

    * added full compilation to new ccnet toplevel

    * lookup scoping draft

    * There's a bug

    * buggy version of struct access implemented

    * sdecl and vdecl as part of scope

    * Fixed codegen lookup bug

    * fixed access of struct elements

    * working on expr
```

```
    * Moved func scope from expr to stmt param

    * merged and began statements

    * sarraylit draft

    * Scall impl

    * function call implemented

    * Implemented Newable

    * added array implementation

    * support for arr_impl

    * fixed typo

    * index on will-codegen-new: bf13497 merged

    * incomplete work on strings

    * removed expression index from sast

    * fixed struct member error for string printing

    Co-authored-by: William Oseghare <osegharewilliam@gmail.com>

commit 25a35840bee6c97fee4cc92d50978c98006139df
Merge: 2361148 f8eb2b5
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 10:51:27 2021 -0400
```

```
    Merge branch 'will-stash' of https://github.com/Bruk3/C-net into
will-codegen-new

commit f8eb2b531f4bd6f6be09b2a3860e1f5e9a63579b
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 10:50:07 2021 -0400

    fixed array literals

commit 23611488b3ed5dd5e997dab9aa80586eb782e6ff
Merge: 48c2fa6 547d9ab
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Wed Apr 21 08:42:16 2021 -0400

    Merge pull request #129 from Bruk3/will-stash

    Will stash

commit 547d9ab8412b7f89bcb8d593b16ab7ec3f18bd69
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 08:37:56 2021 -0400

    fixed typo

commit 919d7257bc874fa0b2e3fd3f41fa028fa9c6da32
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 08:31:17 2021 -0400

    quick fix for pretty printing a struct

commit 3f24208aec4195ea1625afbafb4485da9311af35
Merge: 65c2875 48c2fa6
```

```
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Wed Apr 21 02:19:28 2021 -0400

    Merge branch 'will-codegen-new' into will-stash

commit 48c2fa6cee134189f4032de8f40015bebabe2040
Merge: d5a28e2 1ef5604
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Wed Apr 21 02:12:32 2021 -0400

    Merge pull request #128 from Bruk3/main

    Pull request

commit 65c2875d99fa8db3c2d3a6d18c923dec502b3886
Merge: c3844a3 1ef5604
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 02:10:55 2021 -0400

    Merge branch 'main' of https://github.com/Bruk3/C-net into will-stash

commit c3844a364707652e82feeeedaa986f7448dd75a4
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 21 02:05:19 2021 -0400

    rewrote new and delete to handle strings in a struct

commit 68a7e4bd467bd1215e3a7df5c34deddee8c7e7cb
Merge: 9e653a1 d5a28e2
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Tue Apr 20 20:05:37 2021 -0400
```

```
    Merge branch 'will-codegen-new' of https://github.com/Bruk3/C-net into
will-stash


commit 9e653a1e6806a2e70e53fa0d790b252e78d5a769

Author: William Oseghare <osegharewilliam@gmail.com>

Date:   Tue Apr 20 20:04:03 2021 -0400


    codegen testing


commit 1ef56042f4e89d4e46c82ec795f27cd4a3954fbe

Merge: d4eda73 ccf21a0

Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>

Date:   Tue Apr 20 14:22:29 2021 -0400


    Extra byte allocation and more changes in the string library #123


commit ccf21a06cec0e22edea4054c3c6bd3c421252376

Author: William Oseghare <osegharewilliam@gmail.com>

Date:   Tue Apr 20 14:13:16 2021 -0400


    cnet_read_until now returns empty string instead of NULL if it couldn't
read anything


commit 9777e0308e8487350fa61acca9a983136c1526f7

Author: William Oseghare <osegharewilliam@gmail.com>

Date:   Tue Apr 20 14:11:23 2021 -0400


    Made cnet_strncpy return string instead of void


commit c98df5ae619c2c577878d07d20616c347fd9cda7

Author: William Oseghare <osegharewilliam@gmail.com>

Date:   Tue Apr 20 14:06:30 2021 -0400
```

```
    Fixed bug from null-terminating an empty string

commit 5c6b402a511a6ce22cc281d5f4e389faa8aa98dc
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Tue Apr 20 14:03:07 2021 -0400

    quick fix for cnet_nread. Making sure buf_size is never greater than the
number of chars left to be read.

commit d4eda73b74f3f3b53baec7fc0a8b73fa4cb14fdf
Merge: fb2a6fc a60868c
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Apr 20 13:14:27 2021 -0400

    More semantic tests

commit a60868cba4122417db2845e24b59ade22b19337c
Merge: bf15bab fb2a6fc
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Apr 20 13:04:59 2021 -0400

    Merge branch 'main' into rsemant-test

commit 0a86c3bc201d976b8e59ced554e6bede506eac30
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Apr 20 11:56:17 2021 -0400

    Extra byte allocation for null terminating strings on demand

commit 27dc54582f42f9fc138685b17b445a203a90c11e
Merge: bf13497 11517d2
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 22:55:24 2021 -0400
```

```
    WIP on will-codegen-new: bf13497 merged

commit 11517d27bd9348c72424c176352784f31e8844f1
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 22:55:24 2021 -0400

    index on will-codegen-new: bf13497 merged

commit d5a28e2f9f2922fa18ee851e68db438647234bff
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 22:37:09 2021 -0400

    fixed typo

commit 2cc9402711669928bdffc70297aa6bb33a180d11
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 22:28:04 2021 -0400

    support for arr_impl

commit 015114681c3308a8c33e313bf126de99d88d85a0
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 22:27:32 2021 -0400

    added array implementation

commit 8c73f807e75c40bb6d6a5ecc148ab1785b5c8a1d
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 20:27:12 2021 -0400

    Implemented Newable
```

```
commit bf134979f08a7dbc0c9b338cbe0a8582771ad409
Merge: 76467ac c3db206
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 20:03:48 2021 -0400

    merged

commit c3db206446ef17b879f4c81d75e4bec22b150ddc
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 19:26:39 2021 -0400

    function call implemented

commit 722853c0763685a2eec70bbb6c25145271bf0d7b
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 18:36:46 2021 -0400

    Scall impl

commit fb2a6fce709d56ee62ee762df3cf7abd1d5e0574
Merge: 3f7f9a8 4cf2e68
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Apr 19 18:36:39 2021 -0400

    Prefix user functions with `user_` except main #122

commit 4cf2e68180395d3560347ffc991aabcf17b8ff38
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 19 18:36:01 2021 -0400

    Add test for unrecognized func
```

```
commit 8df514ea05f406bfe76ca5d76564f0b1bb93b660
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 19 18:26:58 2021 -0400

    Remove prefix from main

commit b90056e98267bed381dbbb31e80419b21be0c125
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 19 18:15:22 2021 -0400

    Prefix user functions with user_

commit 7a7395006600cb81bf738508bb8eb7e7afc598c1
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 18:00:22 2021 -0400

    sarraylit draft

commit 76467ac060956513f117c5f02c4c8e2474a495c0
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 16:27:36 2021 -0400

    merged and began statements

commit 23b9e2a79f6ac263011af3db715462b02a9f05c8
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 15:45:53 2021 -0400

    Moved func scope from expr to stmt param

commit a77f94a50989fc1ee7b64016a540f20cb290fcc2
```

```
Merge: ed12e05 2c9c77f
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 15:21:56 2021 -0400

    Merge branch 'will-codegen-new' of https://github.com/Bruk3/C-net into
will-codegen-new

commit 2c9c77fc88f363f383fe60d996382e44ce5a19ab
Merge: 38943ce 1be2d61
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 15:20:13 2021 -0400

    merged with main

commit ed12e0510a8426dfe9054c88de046c10ab033626
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 15:19:37 2021 -0400

    working on expr

commit 3f7f9a8982b199d919ffcba7d80ca04078a262b1
Merge: 1be2d61 9511b28
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Apr 19 15:13:43 2021 -0400

    Parse escaped char literals properly #121

commit 38943ced2c8df97ea5b80c30b2211372631be050
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 15:03:54 2021 -0400

    fixed access of struct elements
```

```
commit ffb378181963d8c8e4c711ccc0ccc0931a298cae
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 14:30:24 2021 -0400

    Fixed codegen lookup bug

commit 1be2d61714e35a0d6800b6d6a07806fb3399686a
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Apr 19 14:06:39 2021 -0400

    A better toplevel that can do everything in one place (#119)

    * added a more flexible python toplevel

    * added full compilation to new ccnet toplevel

    * replaced uses of old top level with new one

    * fixed shebang line and made output executable for binaries

commit eb697fe865fa533b088cf4a0bea9753c68cbfa00
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 13:36:20 2021 -0400

    sdecl and vdecl as part of scope

commit 9511b28db87713fe83bb128a3dd2a1493292e947
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 19 13:14:31 2021 -0400

    Parse escaped char literals properly
```

```
commit af9c57dff19e534acf210bb77be599c1b1014f1a
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 19 12:42:12 2021 -0400

    buggy version of struct access implemented

commit f97af2a0f05c1c85a4ba06bb4c3cc1364f385a4c
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 09:10:23 2021 -0400

    There's a bug

commit f25f9719b02b2fc0c29f7b7b635f9539a2261bfb
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 19 08:25:13 2021 -0400

    lookup scoping draft

commit 34b7d05e21211adece686f098df01a7258af5369
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 18 18:37:25 2021 -0400

    added full compilation to new ccnet toplevel

commit 95e0cc4aad7f78849646c08d7d8ba7dc349160ba
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 18 17:22:08 2021 -0400

    added a more flexible python toplevel

commit bf15bab504838eafafc246564b9a11b594c1387e
Author: Rediet <rsb2179@columbia.edu>
```

```
Date:   Sun Apr 18 12:10:06 2021 -0400

    fix merge conflicts

commit b2a611f06d4a66a12092cd8c70055584754d018a
Author: Rediet <rsb2179@columbia.edu>
Date:   Sun Apr 18 12:09:48 2021 -0400

    fixed merge conflicts with main

commit d4b19eaa43747f722146f392e486e4850a2c010e
Author: Rediet <rsb2179@columbia.edu>
Date:   Sun Apr 18 11:24:45 2021 -0400

    fixed return error statement in var assignment

commit 5fe41649746a0cce8b4b3cac2dcae38bdbe1280c
Author: Rediet <rsb2179@columbia.edu>
Date:   Sun Apr 18 10:50:35 2021 -0400

    fixed error statement for return fail cases

commit fe6ed21ac8c1921202f838fde9ec28d75533d4a7
Author: Rediet <rsb2179@columbia.edu>
Date:   Sun Apr 18 10:39:53 2021 -0400

    removed resolved fail case from semant test

commit 4baadcf96417b5b07e11f88212df8af4afa3a867
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Apr 17 10:18:13 2021 -0400
```

```
    support for struct definitions (including the built in ones) in codegen
(#118)

    * added beginnings of struct typing to codegen

    * a bit more robust checking for built in functions

    * removed casting file <-> socket in built in calls

    * added stub definitions of stdin and stdout

    * implemented consolidated struct support to lltyp_of_typ

    * added support for a struct containing a reference to another struct of
its own type

commit 476290b39690f5307ca53a7bb1b57df8456439aa
Merge: abe4365 c99098a
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Wed Apr 14 14:31:59 2021 -0400

    Merge pull request #116 from Bruk3/kidus-builtins

    built-in functions and variables

commit c99098aa10d72e915a42ca1da563f8a6f27db777
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Apr 14 12:21:11 2021 -0400

    added stdin/stdout as global variables that cannot be defined

commit 2504d0e2166da2d6a3a6d8e1da4c68c0b5f534fb
Author: Kidus Mulu <km3533@columbia.edu>
```

```
Date:   Wed Apr 14 12:06:11 2021 -0400

    added prototypes for built in functions

commit abe43651ad87bc235fa56f9c795296b9c746b246
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Mon Apr 12 20:48:19 2021 -0400

    Update utils.c

commit a7db0df97c8b703dbeeb970edc1e666600a8aa93
Merge: 51a0704 21c9fa7
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Mon Apr 12 20:44:59 2021 -0400

    Merge pull request #117 from Bruk3/string-stdlib

    String stdlib

commit 21c9fa7f73a8cda3f968dbab78a6c52332996add
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 12 20:06:40 2021 -0400

    Commented out valgrind

commit 5592f82fe1a688faa6fde11631c01ffef7201823
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 12 18:01:21 2021 -0400

    Stdlib complete (need more tests)

commit 0b44878486104ef0030970a8d742bbe9ed70ab6a
```

```
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 12 15:45:16 2021 -0400

    added signatures for built-in functions

commit 51a07046a13cc1d5ba2d1398133ca0b399b25082
Merge: cade782 bff5010
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Apr 12 15:17:49 2021 -0400

    Merge pull request #112 from Bruk3/kidus-string-collection

    string automatic collection and other updates

commit bff5010c9a2a3280d6763d6b59d48d4bba2343c4
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 12 15:13:15 2021 -0400

    fixed failing test cases to match output

commit e5fd05301f93f4beabba0700bd7d471f2bc39211
Merge: fab3a89 cade782
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 12 12:00:17 2021 -0400

    Merge branch 'main' into kidus-string-collection

commit cade782d74662204be7403d184f65a482837cb94
Merge: 4fd0c5d b88e935
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Apr 12 11:48:47 2021 -0400
```

```
    Check prototype of function main #105

commit b88e9355a570ea89f22902ef7a1e1379fd249161
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Apr 12 11:40:57 2021 -0400

    main protoype only accepts string array. Fix all previous tests

commit fab3a89db9eaee0850f2a157ef72ba18f10e9d16
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 12 10:59:26 2021 -0400

    added char type to char literals and modified some tests to match current
output

commit 08a784861f43002c343297b15b66b2d8a987a6e0
Merge: 98640d9 d36816c
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Apr 12 10:38:58 2021 -0400

    merged with main

commit 5a8c64f263c89952ccc0a3f077d333dfd6fff6d7
Author: Rediet <rsb2179@columbia.edu>
Date:   Mon Apr 12 00:43:37 2021 -0400

    adjusted for assignment semant test

commit 4fd0c5d82e6c1d724297327d7b897086c66cde02
Merge: d36816c 5fc5f7a
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Sun Apr 11 13:39:04 2021 -0400
```

```
    Merge pull request #114 from Bruk3/Kingsley-add-semant-tests

    Kingsley add semant tests

commit 5fc5f7ab910f233c4d65f218c132c44dd14196cc
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Apr 11 13:35:02 2021 -0400

    clean directory

commit 8b6f59729007fa5b592740b39d562c03cf27f31a
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Apr 11 13:29:18 2021 -0400

    Add illegal str ops and basic arr checks

commit 98640d9fe26593a6e63469643cba53bb7a6edf14
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Apr 11 10:18:37 2021 -0400

    updated return statement checking and finished up global variable
assignment

commit b42aaabc6f3169810f70705fa96dcc004d182e18
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sat Apr 10 22:14:03 2021 -0400

    all string functions tested

commit 241a15ca30dee45dbb61586972793eb222fa2e23
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sat Apr 10 22:02:55 2021 -0400
```

```
    added more tests

commit 5e46d9f3a5f87fcd50f5df0a33d972a3f8d0b5fd
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sat Apr 10 19:26:46 2021 -0400

    added client socket creation

commit fcda99f332670f46a31961163cb9e157710777bb
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Apr 10 13:29:47 2021 -0400

    added return value copying for string returns

commit d36816cb87e2c43a5456ae220db36e396dc30607
Merge: 0402d1e 764212f
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Fri Apr 9 23:36:04 2021 -0400

    Merge pull request #109 from Bruk3/Kingsley-add-semant-tests

    Kingsley add semant tests

commit 764212f0da74ffb0d5a8fdcb28a2da09496acc2d
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Fri Apr 9 23:31:10 2021 -0400

    Add basic scope tests

    Check blocks and function param scopes

commit c1546732a7853b913277f41cb58b36228a430838
```

```
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Fri Apr 9 23:12:27 2021 -0400

    Add basic scope tests

    Check that c-style scoping works with blocks and in function params.

commit f363c915570c0d4db711b656d8f0a662c1ca42ae
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 16:55:44 2021 -0400

    fixed some bugs

commit d4bf964896e8a15a05a94451ab2f760112dc8bea
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 14:49:59 2021 -0400

    added more tests

commit 4f120a510f3fca32fcfa639483e550249cec64c1
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 13:09:53 2021 -0400

    added new io test

commit 8210a9ce86e2f67c67de078739d17f48ab6acf43
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 11:25:20 2021 -0400

    Socket io first draft

commit 24f882b9e22a88290796b347cafa80a83abcfe10
```

```
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 08:52:55 2021 -0400

    test2.c not working yet

commit b3a1ab6156cbc739fd14d140466ff7533fb36736
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 9 08:52:30 2021 -0400

    refactored more code; working on socket creation

commit a45d41ff635b9321ccffb9371bda90f1a4565f63
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Thu Apr 8 22:26:43 2021 -0400

    refactoring

commit 7f1eb008e6fcddc626e04cb6a815667ce5a3d6ed
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 8 17:02:16 2021 -0400

    check protoype of function main

commit 0402d1ea8b858234387571d5924a577e0a9516ad
Merge: 1512210 23de9d2
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Apr 8 15:08:58 2021 -0400

    Type checking for delete and regular expression fix for strings #103

commit 23de9d245dab69750761887d1a2abe08603ed184
Author: Bruk Zewdie <bbz2103@columbia.edu>
```

```
Date:   Thu Apr 8 15:06:17 2021 -0400

    prevent new line matching for string literals

commit 8b0915dadcfd8fe7d93fba0554dee0ea0e9aa83d
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 8 14:48:39 2021 -0400

    Fix regex for string literals. Allow Plus for strings

commit 651ae2b7580569581781a78615801cf52338383b
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 8 14:19:49 2021 -0400

    Type checking for delete stmt.

commit 5da358d2307e9740826fafdc5ce92d4fc674f29f
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Apr 8 13:37:10 2021 -0400

    finished first draft of string collection

commit 579df7083d7f9f34c318bf66aa258968b4556dc5
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Thu Apr 8 10:00:13 2021 -0400

    refactoring some code

commit 5e2312b200f64163e5926cb81125d7ea783f3ec3
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Apr 8 08:54:48 2021 -0400
```

```
    added working prototype of string handling for function calls

commit bb3c40d8dab89a977c5e180784f144bbfc3a3294
Merge: 3cde63f f897ccb
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Thu Apr 8 08:52:18 2021 -0400

    Merge branch 'string-stdlib' of https://github.com/Bruk3/C-net into
string-stdlib

commit 3cde63ff592f993f50f9718cb3cb45f9f24af006
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Thu Apr 8 08:50:54 2021 -0400

    doesn't compile yet (ignore)

commit f897ccb32f9e9c8651c79cc6d281c3cccb662f69
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Thu Apr 8 08:48:57 2021 -0400

    added makefiles for testing

commit e31a3595ad55b32ccaf03720cc2be4f2d02f9053
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Apr 8 00:44:28 2021 -0400

    setup the function to work with temporary strings in expressions

commit 882d9363f718ca2c3a0b24ecaff5da03867e5344
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Apr 7 22:06:37 2021 -0400

    incorporated changes from global variable pr
```

```
commit d806040bcd2e329122cc852c4448f120221d5769
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 7 20:19:20 2021 -0400

    added some tests

commit 7c81497ceb6354eeff5869df43b311d814ca9969
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 7 20:19:06 2021 -0400

    Added stdlib object files

commit d7535bcb977653cd3cd7dc1ed5f053c53d4920da
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Wed Apr 7 20:18:26 2021 -0400

    renamed string functions

commit 77e906963770294bbab0c8a153e64626a958f940
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Apr 7 12:08:51 2021 -0400

    added basic delete statement insersion

commit f6f870c1d78b36d9c91912255e8edf3c99674f27
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Apr 6 00:10:00 2021 -0400

    refactored statement checker to take a record which contains some useful
fields

commit 61ca31457c4d40019823ad2b23a6a702bf038ec5
```

```
Author: Rediet <rsb2179@columbia.edu>
Date:   Mon Apr 5 23:52:53 2021 -0400

    added more fail tests

commit 15122103be63014230b362e606f14506db2a9575
Merge: 3314bf5 835ad7d
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Mon Apr 5 21:03:50 2021 -0400

    Merge pull request #96 from Bruk3/string-stdlib

    Added more string functions

commit 835ad7d805cd0a201c008b1e68798178b3d00c2c
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Apr 5 20:33:13 2021 -0400

    Added more string functions

commit 3314bf5869a3418255c2cde6c0f66c5ba652d010
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Apr 5 09:59:50 2021 -0400

    Semantic checks for array indexing (#92)

    * added output for scope test

    * added semantic checking for indexing an array

commit 9ece7a515ca2a7fc8696487b30fcbb0c07e1aab1
Merge: 00d19e8 144c070
```

```
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Apr 4 04:54:33 2021 -0400

    Int-Char compatibility and Semantic checking for Vdecl_ass #95

commit 144c0706cc4972de8cfdb92c93531485d4ea0609
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 4 04:45:08 2021 -0400

    Fix array expression bug

commit 588694dec4728d62048ede20b64063718602765c
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Apr 4 04:39:30 2021 -0400

    Vdecl_ass semantic check and fix chars

commit 00d19e8d95b4f8cb35e3e50bead75d5f76fb349b
Merge: 4581cad c6138a7
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Sat Apr 3 08:33:01 2021 -0400

    Merge pull request #90 from Bruk3/string-stdlib

    string stdlib implemented

commit 4581cad12a95ef34a79b20478d857a686a9de95c
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Fri Apr 2 22:21:08 2021 -0400

    support for scoping throughout the semantic checker (#87)
```

```
    * added universal bind_checker with scoping to be used in
statement/structs/functions...

    * added scoping to the check statement function

    * added passing new scope to check_stmt function

    * variable declarations now update the current scope

    * added a folding strategy for checking statements

    * implemented scoping in statements, expressions and function bodies

    * added a basic test case for demonstrating scopes

    * removed unnecessary argument from statement checker

    * consolidated difference bind checkers into the most up to date one

    * fixed small warning

commit fc3aebce5f174905f48220fbfa9e37759caf3607
Merge: 8f748a3 ddd0749
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Fri Apr 2 19:12:56 2021 -0400

    Semantic checking for new array expression (#91)

commit c6138a74b9a98da319bd6385c4577ca9bd09b528
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Fri Apr 2 15:20:23 2021 -0400

    string stdlib implemented
```

```
commit ddd07497e6dd867a352fdd40d5eb99c7e89ae78f
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 1 17:06:40 2021 -0400

    Array literal expression support

    add tests for new array

commit 359b3f003308b77ab8f5502f1d849239b07b51ae
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 1 14:38:53 2021 -0400

    fix delete statement

    wip: new Array semantic checking

    Add parser tests for array declaration

commit 8f748a39700e237da663f640740d4b41a620e371
Merge: 6308fae 7b606d8
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Thu Apr 1 17:43:49 2021 -0400

    Merge pull request #79 from Bruk3/Kingsley-add-semant-tests

    Kingsley add semant tests

commit 6308faea316263c435b3eaeaa99eb8ee7dbef770
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Apr 1 14:35:01 2021 -0400
```

```
    Kidus global assign (#81)

    * moved semantic error to sast and added defuault value for unassigned
global variables

    * implemented compiletime resolution of global variables for integer types
and added framework for other literals as well

commit 3e5f1ef04a5cc7a2467a1230f7d2b36f8fe57f0b
Merge: 38372af b35657c
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Apr 1 14:24:30 2021 -0400

    Semantic checking for new struct expressions

commit b35657c3c3b1548279293d2f05b86616007223b5
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 1 14:20:07 2021 -0400

    Add struct expression assignment tests

commit 3d26cdbf2d35a0bcaaea301f960208450cf0c21d
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 1 14:11:41 2021 -0400

    lint

commit 686217637c33e7a490499a8c7c4e72a05e84fa3c
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Apr 1 14:09:46 2021 -0400

    Semantic checking for new struct expressions
```

```
commit 38372af977e832d969c0e60b4a5989a957f4142d
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Apr 1 08:33:19 2021 -0400

    added basic semantic checking for structs and added some fail tests (#77)

    * added basic semantic checking for structs and added some fail tests

    * added checking of struct members that are structs and added some test
cases for that

    * added checking types for struct member access and added basic checking
for that

    * added checking types for struct member access and added basic checking
for that

    * added type checking of variable declarations and some tests for that

commit 7b606d8b6c63cf6129459a7d5895d39d276d0d6e
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Thu Apr 1 07:39:59 2021 -0400

    Add more test-*.cnet

commit 3264cfe8b3550d6a6ec1ba9038076e479f0809da
Author: Rediet <rsb2179@columbia.edu>
Date:   Thu Apr 1 02:11:53 2021 -0400

    add more fail case semant tests

commit 8ae315b0873e976b49fa024abcb07f2d0ea5f3b0
Merge: 310c9d8 9373fd9
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
```

```
Date:   Wed Mar 31 13:58:41 2021 -0400

    More parser tests

commit 9373fd9d76c18fcf9e218a340d7eae07e2b632e9
Author: Rediet <rsb2179@columbia.edu>
Date:   Wed Mar 31 11:14:49 2021 -0400

    fixed error messages for parser tests

commit 76e9e2a61da835341ab358f386e8b54f24c5f52f
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Wed Mar 31 07:27:28 2021 -0400

    Add fail semant tests

commit 05e4dcf20de829c10e23b5fc57ba51cee978a549
Author: Rediet <rsb2179@columbia.edu>
Date:   Wed Mar 31 03:35:15 2021 -0400

    rebase from main

commit 310c9d8f11e0df2fd38f1e6bfc85ff53fbffd8c0
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 30 16:23:33 2021 -0400

    added an easier/extendable interface to semantic errors (#78)

commit 32eded4566480d2f07ee3b4f1027a1dc6888cd73
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 29 20:53:35 2021 -0400
```

```
Added multiple indexing and assigning to index (#68)

* allowed for treating an index as an id

* added tests for index assignment and multi-level indexing

* Fix CI to support llvm (#72)

* fix CI to support llvm

* try old minimal

* add install commands

* update ubuntu version

* use llvm-action

* comment out notification and echo yes

* Use caching and redirect Y through yes

* Added yes flag

* Add caching and remove redundant ocamlfind

* remove caching

* Update _tags to include llvm

* Uncomment notification and clean workflow
```

```
* Update PATH with path to llvm

* create symbolic link and remove caching and fail fast strategy

* remove debug statement

* Fix typo

* comment out llvm installation to check if it works

* New workflow with llvm is functional

* added support for file type in scanner/parser/ast (#71)

* added support for file type in scanner/parser/ast

* uncommented llvm line in _tags

* added more helpful info to scanner errors (#69)

* added more helpful info to scanner errors

* uncommented llvm line

* fixed test cases to match new error output

* added a parser error case for runtests to succeed

* Add support for complex C float

* Remove _tags from .gitignore and remove comments
```

```
    * lint

    * If/else handled in semant. Semantic checking part of tests

    * allowed for treating an index as an id

    * added tests for index assignment and multi-level indexing

    Co-authored-by: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
    Co-authored-by: Bruk Zewdie <bbz2103@columbia.edu>

commit f1bd0f7d9981f21bec1c6a984e5bf55feb9a3088
Merge: f3bd2a7 0108363
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 29 20:39:19 2021 -0400

    Add support for conditionals in semant.ml. Semantic checking added in tests

commit 01083632c2737ae1a46825b469001a8f3947089a
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 29 20:32:54 2021 -0400

    If/else handled in semant. Semantic checking part of tests

commit f3bd2a7b15d1c9c564d1c0bfca0d7338ac082b31
Merge: 55298d9 9b4ed57
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 29 13:35:02 2021 -0400

    Add support  for complex C floats

commit 9b4ed57b3d702a541b55c8f0099788d73cc03bd2
```

```
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 29 13:27:52 2021 -0400

    lint

commit af9213119a52755c3692f4dbb6556fe125c3b195
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 29 13:25:48 2021 -0400

    Remove _tags from .gitignore and remove comments

commit a5ce3da658e49713a07a167e04b987d46fad0af5
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 29 13:24:09 2021 -0400

    Add support for complex C float

commit 55298d960f8a7df195b07058da3a51b7c8658697
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 29 10:19:24 2021 -0400

    added more helpful info to scanner errors (#69)

    * added more helpful info to scanner errors

    * uncommented llvm line

    * fixed test cases to match new error output

    * added a parser error case for runtests to succeed

commit 3fb2f34796b8705c17debe3e868f647f60d5f1c2
```

```
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 29 09:17:58 2021 -0400

    added support for file type in scanner/parser/ast (#71)

    * added support for file type in scanner/parser/ast

    * uncommented llvm line in _tags

commit 45b23a9a2585dd18fb9417870046446de1e09ca6
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 29 03:49:01 2021 -0400

    Fix CI to support llvm (#72)

    * fix CI to support llvm

    * try old minimal

    * add install commands

    * update ubuntu version

    * use llvm-action

    * comment out notification and echo yes

    * Use caching and redirect Y through yes

    * Added yes flag

    * Add caching and remove redundant ocamlfind
```

```
    * remove caching

    * Update _tags to include llvm

    * Uncomment notification and clean workflow

    * Update PATH with path to llvm

    * create symbolic link and remove caching and fail fast strategy

    * remove debug statement

    * Fix typo

    * comment out llvm installation to check if it works

    * New workflow with llvm is functional

commit f7cd3fbe0964deab014103daa602c98469f0a6d6
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 24 20:23:39 2021 -0400

    Update README (#67)

commit 8eeb2e018270116dfe097504e849b1cfed949bbc
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 24 20:12:49 2021 -0400

    Fix all warnings (#66)

commit 32adce54bb82339e1144b314f43cd974b1e4c5b0
```

```
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 24 18:38:14 2021 -0400

    Hello world integration (#65)

    * Codegen first draft

    * copy semant.ml from microc for starter

    * wip: semant.ml implementation started

    * finish modifying expressions; wip statements

    * Squashed a sfew syntax errors.

    * Squashed more bugs;
    Added new target in Makefile;
    Updated cnet.ml

    * most recent changes for static semantic checking; pushing to a new branch
from here

    * more bugs

    * removed printbig references

    * got hello world working on a hard coded sast

    * got hello world working through sast

    * fixed new line issue in println and added some testing; cleaned up
cnet.ml
```

```
    * added top level compiler to Makefile

    * added testing for integration tests

    * check if llvm compiler exists first

    * Rename llvm compiler. Minor fix in libcnet. Makefile updates

    * whitespace linting

    * got proper return value from write for println

    Co-authored-by: William Oseghare <osegharewilliam@gmail.com>
    Co-authored-by: Kidus Mulu <km3533@columbia.edu>

commit 292c6ab6f0876b3c3e8d7edbfe0ddf652aa56e94
Merge: f199820 9d8d0d4
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 23 20:44:48 2021 -0400

    Merge pull request #64 from Bruk3/kidus-general-improvements

    added general improvements

commit 9d8d0d40eb4a72c63a62eaa13808dbe26f492694
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 23 20:37:41 2021 -0400

    added general improvements

commit f1998201fb3f27141a25eb56105f2796b2fb05ea
Merge: 1b898a4 67fa3af
```

```
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 23 20:11:36 2021 -0400

    Merge pull request #63 from Bruk3/bruk-semantic-checking

    First implementation of semantic analyzer

commit 67fa3afd30915c08c0aac85fde0810f126e51700
Merge: 0f4e14b 1b898a4
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 23 17:06:57 2021 -0400

    Merge branch 'main' into bruk-semantic-checking

commit 0f4e14b58857c1c0ca14b2c1eaf6051658ac61f6
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 17:05:38 2021 -0400

    update test src file

commit c49a7519b4881e7496c6582c7c93a78382487e60
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 15:56:02 2021 -0400

    fix Vdecl_assign

commit 2c6d57eb9712b80196d98c3484dbf1a40537fa56
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 15:48:07 2021 -0400

    local variable declaration and assignment works
```

```
commit 555f3f5022d757a856e51dcfad91296a2995fb4c
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 15:37:23 2021 -0400

    Add locals field to funtion

commit a308e12fa897c4a5ce5c493422ec14fa41d76079
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 15:18:04 2021 -0400

    fix variable declaration

commit 61739ed6fcb36ac4e5b0b91c491f00629449601b
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 14:56:28 2021 -0400

    revert Makefile and _tags changes

commit a3a6c7c53878c6a096b7debf721c3f9d07b2ef75
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 14:51:27 2021 -0400

    remove printbig.c

commit dc5c59f729296cca30fa35328f14968c605869c3
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 23 14:49:52 2021 -0400

    undo all codegen changes

commit e496fb66838d2c76bdd2b788d5715b8f81b58d83
Author: Bruk Zewdie <bbz2103@columbia.edu>
```

```
Date:   Tue Mar 23 14:44:50 2021 -0400

    basic semantic type checking that passes simple test

commit 1b898a4fcb5ddd441f22d55260109c20bc71229d
Merge: 1ee0810 7859978
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 22 21:13:35 2021 -0400

    Merge pull request #62 from Bruk3/kidus-makefile-update

    added cleaning codegen products in clean rule

commit 1ee08109217ed69d5cb2879099e260b6f3b042d6
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 22 20:28:30 2021 -0400

    Add _tags to .gitignore (#61)

    * Update _tags with llvm package directive

    * comment out llvm directive in ctags

    * Add _tags to .gitignore

commit 41e961f67e08ea1ffec041256c23cba7899f5711
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 20:18:55 2021 -0400

    update cnet.ml

commit 785997831da661d18ccaac49aee490cbd59ea2db
```

```
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 22 20:05:02 2021 -0400

    added cleaning codegen products in clean rule

commit 60712b574bab35372c388a1c4133183757af742d
Merge: 5782409 313e1cd
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 19:52:26 2021 -0400

    Merge branch 'codegen' of https://github.com/Bruk3/C-net into codegen

commit 5782409279dcd03ba694e57a989437c5dbbc2743
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 19:52:16 2021 -0400

    most recent changes for static semantic checking; pushing to a new branch
from here

commit 313e1cd389ed43bcc7bab24415e9bdf3f1cbffc8
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Mar 22 19:46:00 2021 -0400

    more bugs

commit 67ae480e2b79aee00a48742961110528556df7d5
Merge: da2ef03 f710296
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 17:06:21 2021 -0400

    Merge branch 'codegen' of https://github.com/Bruk3/C-net into codegen

commit da2ef0300916909cfe0b14a3f48fd8cba5b5b148
```

```
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 17:04:15 2021 -0400

    finish modifying expressions; wip statements

commit 0b1f336a1a922c3b03c36289d4f36313949b0009
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 22 16:15:15 2021 -0400

    wip: semant.ml implementation started

commit f710296893fe0ae3c4ec815dec1a4ce3419b6151
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Mon Mar 22 10:14:51 2021 -0400

    Squashed more bugs;
    Added new target in Makefile;
    Updated cnet.ml

commit 4e9a195e326e05a3fde0afb19df5769d2bec70b3
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Mar 21 21:32:35 2021 -0400

    copy semant.ml from microc for starter

commit ec1175911b01e608e39c6a5283b25a5092a0658c
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Mar 21 17:08:38 2021 -0400

    Squashed a sfew syntax errors.

commit 567b99232dff73d9b033ea6dc1d977852bfab7c0
```

```
Author: William Oseghare <osegharewilliam@gmail.com>
Date:   Sun Mar 21 13:10:38 2021 -0400

    Codegen first draft

commit e7d6c611e955b63c9737518ae24515779f0258ef
Merge: eaa2915 240a198
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Sun Mar 21 00:05:05 2021 -0400

    Merge pull request #46 from Bruk3/Kingsley-add-parser-tests

    Kingsley add control flow tests

commit 240a19835aeba5210c6bbde51a285684a3e9a47d
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 21:13:59 2021 -0500

    Add nested-mix test

commit 8adffbb265444a49b4cf91634ddb77d993b680af
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 21:01:12 2021 -0500

    Add break-continue test

commit 0e53c08ff15985de0f86bf25664b76a83871f382
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:51:40 2021 -0500

    Add infinite for test
```

```
commit 630c4c5741dd6aab03664fdd405556531132bd8b
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:46:23 2021 -0500

    Add nested for test

commit 16750aa4775708af3c10d320e1274bdaacf14f5b
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:43:19 2021 -0500

    Add nested while test

commit 2cb2449e66884ab71abf2c3d517eaeb0a655ce6d
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:37:41 2021 -0500

    Correct test-vdecl file name

commit c027dc0e202f650be19e654a7c1b5ee7f6078dc9
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:37:18 2021 -0500

    Add nested ifelse test

commit 70f6837b73e4540a8a14605b9263be219969a3e3
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 20:26:03 2021 -0500

    Update test file name

commit 224b33565db7079da0f28e4ce7a0fbabe7b18d69
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
```

```
Date:   Sat Mar 13 20:23:11 2021 -0500

    Add case for if w/o else

    Adding this case fixes the issue of an unnecessary else statement being
added to a if statement  without an else statement

commit 52269f138329591a6cb85dc43d411f6614b6df54
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Mar 13 19:23:25 2021 -0500

    Add if-else-while-for test

commit eaa291544eee539f6e18c9bb90c7453309897672
Merge: 09ce3e2 9939f3a
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Mar 20 22:20:58 2021 -0400

    Merge pull request #54 from Bruk3/kidus-better-errors

    Better error messages when there is a parsing error

commit 9939f3a7db3036e6deda54a0f122180d0b383447
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Mar 20 21:33:04 2021 -0400

    cleaned up cnet.ml

commit bca33125567bd52a6b56fb8c2421882fd722e760
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Mar 20 21:21:12 2021 -0400

    finished whitespace counting and updated Makefile and error printing format
```

```
commit 6de780deaca53cf0d560b6c9fcbd56e0f7dab8ea
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Mar 20 20:53:34 2021 -0400

    fixed line numbering in all cases except else if clauses

commit e608b32aa646cfc397bbb602b2cd6591a1070c87
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Fri Mar 19 13:22:18 2021 -0400

    used module name to remove compilation warning

commit 6ccfe231e75c585044377ea2ee17a4ee9a0b622e
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Fri Mar 19 13:17:10 2021 -0400

    simple line and token error reporting for parsing errors

commit ea32e528d1a749ed09c6b9d29b32dc974aef51f2
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Fri Mar 19 12:57:58 2021 -0400

    added line numbering and buggy version of error reporting

commit 09ce3e26ae4d0322332de5471647fa9877094bfc
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Mar 20 21:59:05 2021 -0400

    Revert "Expressions test (#57)" (#59)

    This reverts commit 039b57f4c661e023effb6852f9e59c9b08d1b015.
```

```
commit 039b57f4c661e023effb6852f9e59c9b08d1b015
Author: William Oseghare <48426646+MaverickMiles@users.noreply.github.com>
Date:   Sat Mar 20 21:56:28 2021 -0400

    Expressions test (#57)

    * parser test for expressions;
     support for multidimensional array;
     added brackets for expr pp;

    * updated assign_operator expr matching

    * updated test-expression tests

commit d785a77be22cf41ef311b1099f0cb22a8315c973
Merge: 0a0c8ec 18b9307
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Mar 20 21:55:21 2021 -0400

    First draft for sast to use for hello world deliverable

commit 0a0c8ec45535e38b1d94359304a6322e7c7fd00e
Merge: d22003c f1482f5
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Mar 20 21:39:15 2021 -0400

    remove newline in struct pretty printing and add tests

commit f1482f5ae969bd2866b00570cc2ecf4577fe6d1b
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sat Mar 20 21:32:37 2021 -0400
```

```
    remove newline in struct pp and add tests

commit 18b9307f74d8b16ca6dca9fc914919ca14ec3391
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Mar 17 12:10:28 2021 -0400

    first draft of sast

commit d22003c64552bec363da7d7e836cf47c7bd5d16b
Merge: 2f18355 6c83ccc
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 16 15:48:39 2021 -0400

    Merge pull request #50 from Bruk3/kidus-array-fix

    fixed reversed reading of array literal elements

commit 6c83ccc21c0014a27ff91348e94ce252ff133360
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 16 15:35:48 2021 -0400

    fixed reversed reading of array literal elements

commit 2f18355992ceb45d014e0b6dae6e722ca1af536e
Merge: 8d9dee1 afae382
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 15 18:19:10 2021 -0400

    minor Makefile fix. Remove parser.output during clean

commit afae382184ab8d801e4a3e479f7d0780766ee1b2
```

```
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Mar 15 18:00:27 2021 -0400

    rm parser.output in make clean

commit ec3d84efd6864659f8b01a7e3f3692dd0efca328
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sun Mar 14 19:32:40 2021 -0400

    added first draft for sast types

commit 8d9dee12f9584ca8e0ab3dabd1c4d20bdc1e59fc
Merge: f3b0d68 9ed4f3b
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sun Mar 14 19:28:48 2021 -0400

    Merge pull request #45 from Bruk3/kidus-assign-fix

    Associativity fix for assignment

commit f3b0d68ae0289c87a0277ef41cf5aea1f39c9378
Merge: f4e793b d5bd9e9
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Sat Mar 13 21:09:28 2021 -0500

    Merge pull request #44 from Bruk3/kidus-delete-fix

    moved delete from being an expression to a statement

commit 9ed4f3b1a6c549d3d3db3c585cdf5c291049ba16
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Mar 13 21:07:48 2021 -0500
```

```
    fixed up scanner a little more and adjusted test files to match

commit be8255ba97d73cd2c810d4c03eb86667a2150ca9
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Sat Mar 13 20:59:34 2021 -0500

    fixed assignment precedence bug and cleaned up scanner a little

commit d5bd9e96d7f3ddae0591c4578edc8e5f439beeff
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Fri Mar 12 20:35:05 2021 -0500

    moved delete from being an expression to a statement

commit f4e793b8ecb8bda7afdb37a492faf14f87d68fc4
Merge: 7f4d510 06524ab
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Mar 11 17:53:39 2021 -0500

    Merge pull request #42 from Bruk3/kidus-add-breakcontinue

    Added break and continue support and also fixed fails due to whitespace

commit 06524abbe31492fb198338f908fc991f8f960c08
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 11 17:40:57 2021 -0500

    fixed test fails on whitespace

commit 1c9f253f58cb8fcd50befbbe149fcfef4f84f67d
Author: Kidus Mulu <km3533@columbia.edu>
```

```
Date:   Thu Mar 11 17:33:41 2021 -0500

    added support for break and continue to parser and ast

commit 7f4d510f52b06f53823526c02f8184250e5b5212
Merge: cf25511 78e5bd2
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Mar 11 17:11:57 2021 -0500

    Merge pull request #39 from Bruk3/kidus-elseif-fix

    Fixed associativity of else if by introducing a separate token for else if

commit cf255115c925169f9d308a6b6ca24cff7f28d02a
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Thu Mar 11 16:07:18 2021 -0500

    Add assing, decl, control flow tests

commit 78e5bd2ead0e8ec83d3fbe95e035fa33266efd30
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Mar 10 20:28:39 2021 -0500

    adjusted whitespace error in the func testcase

commit af1b040ac16a31073ece6b0dc46f3872a9319b66
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Mar 10 20:26:10 2021 -0500

    fixed else if associativity error

commit ba41b514f90d8f252ac80cf95bcf8732b41858c8
```

```
Merge: 2e5af72 4d494c8
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 10 17:39:52 2021 -0500

    Test script runs both parser and scanner tests

commit 4d494c8218423af349388a72fb70a1b41d636d41
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 10 17:31:00 2021 -0500

    Support for parser tests. Example tests included

commit 2e5af7217894a07ec680bffb36d57ccb74151581
Merge: 6c0c835 28d31e5
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 10 17:13:30 2021 -0500

    pretty printing improvements; support for global declaration and
assignments in parser and ast

commit 28d31e53c168b3caaab0c3f6752ae69183dc6f99
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 20:57:03 2021 -0500

    fixed messed up formatting on if else statements

commit ea2aba1377636f61c69c83ce5e7ce738f9a51a9a
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 20:24:27 2021 -0500

    minor updates to Makefile

commit 929867aba7f29ed1c6d81215d3a3c55b6e992e63
```

```
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 19:13:08 2021 -0500

    fixed lack of cnet.native dependency in Makefile

commit 71468989e05298a469e6ba2f9e81a7575c54d0e2
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 18:44:22 2021 -0500

    finished support for indentation in pretty printing

commit 23d4f23cc4d1a8cbe6a0b37daf095a1131c47a3a
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 16:20:35 2021 -0500

    updates to pretty printing; still incomplete

commit 6c0c835ee64aab71fd46036817daf5aa6e8cdebd
Merge: 4ffad66 3f0efc7
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 9 20:04:58 2021 -0500

    Restructure tests and support for -t flag

commit 3f0efc74466e4b6a58b4cd1cc151d8b34c1d9d26
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 9 19:56:18 2021 -0500

    fixed failing test case and clean target in Makefile

commit 169421d765d354c8c26340e92e4873f3aed5d661
Author: Bruk Zewdie <bbz2103@columbia.edu>
```

```
Date:   Tue Mar 9 18:22:55 2021 -0500

    install ocamlfind

commit 7b2c0c2b9c664a1d9e7a56f939e64fcde58f4a23
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 18:12:09 2021 -0500

    Github CI install ocamlbuild

commit bc7c080a7dd8ebaba81ba19e8c8aa84397c831d6
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 17:56:20 2021 -0500

    fix for github actions

commit d6cc32db89f71612d4b796b8c0446b94c51911ec
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 17:49:51 2021 -0500

    fix ci-test target in Makefile

commit 9e036003e801c31e84b2e19fde0076ae017f2d0e
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 17:43:27 2021 -0500

    Makefile update: old scanner tests part pf ci workflow

commit efac05e7c3c9e4d1156c76d438af082c1e47db2f
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 17:43:02 2021 -0500
```

```
    clean scanner and fix TCP/UDP in expected output

commit 979c79ec42408444e55b5b065ded54ed7fa76427
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 9 17:32:38 2021 -0500

    Restructure tests directory, New -t flag for old scanner tests

    To pretty print the tokens (Like the old scanner tests) use the following
    command:

    make cnet.native
    ./cnet.native -t "tests/scanner/<file>.cnet"

commit 4ffad6607ee53b5ea333ba3c7317f47139f80a41
Merge: 9821747 5593530
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Mar 8 17:52:34 2021 -0500

    If-else-if support

commit 55935309393d727b7b860b08f317860593d955f3
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 4 18:18:53 2021 -0500

    removed TCP/UDP tokens

commit 7698ca239a40ded984b3e4ec3130c6f49807a916
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 4 17:17:34 2021 -0500

    fixed non-existing token
```

```
commit 45cc0a405faad1d2ad9d9216a8394bab5482f43e
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 4 17:15:02 2021 -0500

    fixed else if associativity

commit cc8aaadeed9857efb1cddefa2b20078479900c25
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Mar 4 16:41:44 2021 -0500

    remove redundant newline for if expressions

commit 06d1dd5c3cadf0ead3710814ac0caf239786b3a7
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Mar 4 16:00:55 2021 -0500

    fix pretty printing for new if/else if

commit 9821747aca664f744a58fd5aab6d9f45e286bbd0
Merge: 82b0708 116e407
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Mar 4 13:11:21 2021 -0500

    Scanner test

commit 8f6e189cdecaa2eb3d14701dfffb8c0f5c060814
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 4 09:21:08 2021 -0500

    allowed statements with empty expressions
```

```
commit ca5101a5eedc9a9cba0d3f876126804732685afd
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Mar 4 09:05:25 2021 -0500

    added if-else-if support; pretty printing broken again :(

commit 116e4077e0c70cd424725410f0f32d621998e3cd
Author: Rediet <rsb2179@columbia.edu>
Date:   Thu Mar 4 02:28:26 2021 -0500

    removed log and parser.ml from commit

commit 765542a9f478dd38ae67ea99ff071d9233c159c4
Author: Rediet <rsb2179@columbia.edu>
Date:   Thu Mar 4 02:16:50 2021 -0500

    fixed test conflicts with updated tokens

commit 2bcbff96735d70d08eab0c962653f9c07ffc7105
Merge: b96ed8e 82b0708
Author: Rediet <redietbekele@dyn-160-39-255-129.dyn.columbia.edu>
Date:   Thu Mar 4 01:53:15 2021 -0500

    updated scanner from latest commit

commit 82b07080e9f953f6b0d51d65576344b0e4ed66b5
Merge: aa4d052 6fc9948
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 3 17:55:06 2021 -0500

    Pretty printing works!
```

```
commit 6fc9948df0cd9c2695a9cbbc1d8fe46bd28a1e32
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 17:46:41 2021 -0500

    Makefile new targets; Old pretty printer in separate scanner_pp.ml file

commit 40f4d8f3c664401feb18412d35f5694206e3e0cf
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 17:22:25 2021 -0500

    removed old pretty pretting from scanner; parser pretty printing works

commit 4537b2a2daf7d2f006c8912929d0b63e6788b2b7
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 17:21:14 2021 -0500

    fixed all pretty printing type errors

commit c3eb6c9747abd97412efb8c48b6640bbb86ea309
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 14:37:58 2021 -0500

    remove NOELSE from tokens; small fixes

commit 129d92ea6706955005aae5a974326e94b1f7b58f
Merge: 63cdf6a e14fee8
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 14:23:54 2021 -0500

    remove NArray from newable

commit e14fee82e37829fee2c4d9be792c3b99796af8a9
```

```
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 13:45:04 2021 -0500

    delete parser.ml

commit 40642882628869ae2a18eed267d8d1e0c3532fbe
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 13:42:19 2021 -0500

    add toplevel and some fixes to parser

commit 63cdf6a9d766c3c5dbcb02f46dd84fd124555e4a
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Wed Mar 3 12:57:43 2021 -0500

    moved array literals to expression type in ast and changed rule for new;
prettyprint broken

commit aa4d052aaf4336b3f6ef62fa9c68b547de5b5998
Merge: b45a148 e0c0ff0
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 3 12:01:25 2021 -0500

    Pretty-printing functions for type rid and id

commit e0c0ff07ec9274dbe03d74dcf3261d85b8138ff6
Merge: f526970 90467cb
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 11:52:24 2021 -0500

    Merge branch 'pretty-printing-ast' of https://github.com/Bruk3/C-net into
pretty-printing-ast
```

```
commit f5269709e6c24cd302c5c20fd6bada4889f82a77
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 11:52:13 2021 -0500

    Add printing functions for type rid and type id

commit b45a1483835252c216380a06cd6338410f3d1bdc
Merge: 6637730 90467cb
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 3 11:44:19 2021 -0500

    Add pretty-printing functions (#28)

    * Add pretty-printing for expressions, types, and, operators

    * pretty printing work in progress

    * fix type errors in most functions except string_of_func and
string_of_decl

    * fix all compile errors

commit 90467cbb08ac444485d6cb4bd1122cf7c16ed993
Merge: 1137b12 6637730
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Mar 3 11:38:07 2021 -0500

    Merge branch 'main' into pretty-printing-ast

commit 1137b12818f3fd4f9182305c533d0242d58837f7
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 11:34:20 2021 -0500
```

```
    fix all compile errors

commit 7f308e31b607ca37873a9ff696752b9272dd67c6
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 11:27:16 2021 -0500

    fix type errors in most functions except string_of_func and string_of_decl

commit 58a3275729150e0f8e44075666c1e127d064b736
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Mar 3 10:55:48 2021 -0500

    pretty printing work in progress

commit 0a6bad3db6c8dec1d3ab81c9872fd8f9db780652
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 18:56:27 2021 -0500

    Add pretty printing for expressions, types, and, operators

commit 6637730acadf9c51af3c55031343bb9f50d3a275
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 2 17:20:42 2021 -0500

    Make all binary operators the same type (#26)

commit ea48b7464bcd09b0596cddd47ed5accc23b76c15
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 17:13:20 2021 -0500

    Make all binary operators the same type
```

```
commit f763419b139f674ddd3539d38fac5cd8b05909b2
Merge: b62dabf 296e39b
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Tue Mar 2 16:48:39 2021 -0500

    Merge pull request #25 from Bruk3/Kingsley-scanner-test

    Add exceptions for dquote and multiline comment

commit 296e39b18af1e949fe67d8a414f8018503d4e04f
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Tue Mar 2 16:47:51 2021 -0500

    Add exceptions for dquote and multiline comment

commit b62dabf016158744b85d063d62a36c3d100889e9
Merge: ea5ae92 f6a7192
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 2 16:03:18 2021 -0500

    fixed new and noexpr

commit f6a7192723ff3a69e3f9609a017544ee7c03377e
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 15:51:45 2021 -0500

    Add Block, rename Statement to Expr and update Makefile

commit ce1be95aad184986a6ee72a4ac7f742c093bda5b
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 2 15:38:46 2021 -0500
```

```
    fixed new and noexpr

commit ea5ae92a6ed60b8a0844b17c40dd31949fcb1d9f
Merge: 9392360 95757e2
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 2 15:19:12 2021 -0500

    Merge pull request #23 from Bruk3/kidus-parser-actions-enhancements

    finished up actions for parser and edited array type in ast

commit 95757e2ab089703f5ecd967411f37e9c0f59858b
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 2 15:12:07 2021 -0500

    finished up actions for parser and edited array type in ast

commit 9392360edb869b46effdc62b265c1422f9d13c16
Merge: 8cf7b1f f12b088
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 2 14:23:17 2021 -0500

    Add more actions to basic conditionals and loops

commit f12b08865eb37653c110eca25fbf40e9af9b97ec
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 14:16:49 2021 -0500

    Print success in githuc CI workflow

commit 0f8325eecf20b4678ef10e16699895be3c07f902
Author: Bruk Zewdie <bbz2103@columbia.edu>
```

```
Date:   Tue Mar 2 14:16:32 2021 -0500

    Add parser actions for variable declarations

commit ff168060f24b1a7be9bb0b52e367a695f8bc643e
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 14:01:57 2021 -0500

    removed all types from ast. removed newable rule

commit 31ecbaee47c90fc0aaff57b92def6739d0961f69
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 13:56:12 2021 -0500

    Add more actions to basic conditionals and loops

commit 8cf7b1f0a7006de018caedd5676585f3969aa464
Merge: 3d373fe 8a9b653
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Tue Mar 2 13:33:59 2021 -0500

    Merge pull request #21 from Bruk3/kidus-ast2

    Merge the completed ast types

commit 8a9b653ca400e12fd19faa640b59cc60967afba7
Merge: 26947e3 102209c
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Tue Mar 2 13:28:37 2021 -0500

    Merge branch 'kidus-ast2' of github.com:Bruk3/C-net into kidus-ast2
```

```
commit 26947e33603b7d4c29f8c882e292a92e567d2ac5
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 21:00:20 2021 -0500

    finished first draft of ast types

commit b3db91dcbb08e852bcf924a4f34a8a25c5762e29
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 20:23:13 2021 -0500

    cleaned up statement rule in parser and added basic statement type in ast

commit 693522cf62be1c6c592aa3458427c7c4d9412838
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 13:30:21 2021 -0500

    fleshed out expresssion type in the ast

commit 3d373fe21bd083b1d8219e3a7335ddcfed0b0780
Merge: 9bd055f 4a7bffd
Author: Kingsley Neequaye <36528120+king751@users.noreply.github.com>
Date:   Tue Mar 2 13:15:01 2021 -0500

    Merge pull request #20 from Bruk3/Kingsley-scanner-test

    Add remaining categories of tests for scanner

commit 9bd055f01c7b0e853d0d186139dbcef54f1935e0
Merge: 7f54787 e9b2f04
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Mar 2 11:37:55 2021 -0500
```

```
    Add actions to parser

commit e9b2f041a435685aa5d92568e042f0d684fbca9d
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Tue Mar 2 11:30:34 2021 -0500

    Add actions to parser

commit 102209c8e5baa068a9236cb54c67beb039590e0d
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 21:00:20 2021 -0500

    finished first draft of ast types

commit 765f48c97244581386c84ec56c4c5cbee0667cb7
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 20:23:13 2021 -0500

    cleaned up statement rule in parser and added basic statement type in ast

commit b96ed8efca0edf6e16769e36a413bb450ac7cc4e
Author: Rediet <redietbekele@dyn-160-39-255-129.dyn.columbia.edu>
Date:   Mon Mar 1 15:08:12 2021 -0500

    added more test cases

commit 7f547870ce23040f7b0c3102d58571c29330e297
Merge: 5e5732f b5aa593
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 1 14:12:22 2021 -0500

    Merge pull request #17 from Bruk3/kidus-ast2
```

```
    added basic ast.ml; fixed Makefile further and turned structmem to id…

commit 2b9ec82a41cc046e64bb349bd035fc90af583cfa
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 13:30:21 2021 -0500

    fleshed out expresssion type in the ast

commit b5aa5936a0048855a7a75f11b2a0447cd49507ca
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 12:31:17 2021 -0500

    added ast types for operators

commit c4f150abc58d8de9d9014848f0d23721db9e7b41
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 12:01:32 2021 -0500

    added basic ast.ml; fixed Makefile further and turned structmem to id in
grammar

commit 5e5732fa6657fabbe3c815a86293984ee5afb47a
Merge: 55d82e5 98e09c1
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Mon Mar 1 09:57:58 2021 -0500

    Merge pull request #16 from Bruk3/kidus-ast

    Makefile updates for proper dependency and small fixes in scanner and
parser

commit 98e09c11cbde0ce893e86d4403f2f41381cfba4d
```

```
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 09:48:30 2021 -0500

    added ast.mli and dependencies in Makefile

commit 97b8d6eca516fa704a2eed41a9396cbc769419df
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Mon Mar 1 09:32:39 2021 -0500

    resolved makefile dependencies and included parser in scanner; also fixed
some token conflicts in parser

commit 4a7bffd13e1adad39cc49d80c251af68d2a9350d
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Mon Mar 1 05:30:23 2021 -0500

    Add remaining categories of tests for scanner

commit 55d82e5ae48f3c1126a604983007831dd920d510
Merge: 4c9949d 87416fd
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Wed Feb 24 20:14:55 2021 -0500

    Scanner fixes and Partial pretty printing

    Scanner fixes and Partial pretty printing

commit 87416fde4f157cf72ce47e18c1f3d0e5c76d3de5
Merge: d403b0f abf7613
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Wed Feb 24 18:51:10 2021 -0500

    Merge branch 'rbranch' of https://github.com/Bruk3/C-net into rbranch
```

```
commit d403b0f761fb4cfcb1a278bbd04dd555d085c5b1
Author: Rediet Bekele <redietbekele@dyn-160-39-255-129.dyn.columbia.edu>
Date:   Sun Feb 21 16:06:21 2021 -0500

    Scanner fixes and pretty printing

    test case, pretty printing and tokens added

    test case for function scanner added

    Add test for literals

    Add test char literal to literal test

    Change FLOATLIT to float type

commit 4c9949dea3a9f05cde6422eca2700405937d6457
Merge: dafd8fb ea55814
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Feb 22 21:26:20 2021 -0500

    Add .gitignore and fix CI action failure

commit ea558141f533cbe26ccf6ccc1bbfc8aa79ef02c1
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Feb 22 21:20:18 2021 -0500

    Add newline before EOF

commit cc16b95d765e916d3572d339e88a4b4091a13485
Author: Bruk Zewdie <bbz2103@columbia.edu>
```

```
Date:   Mon Feb 22 21:18:35 2021 -0500

    Add .gitignore and fix CI action failure

commit dafd8fbaec10d38d441ed915e9d5ea1eddf9d604
Merge: 4b6ad0d 60bb34d
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Mon Feb 22 20:40:55 2021 -0500

    Merge pull request #8 from Bruk3/kidus-fix-grammar

    Grammar with no errors (Untested)

commit 60bb34db90022dc34393d9c5c0f0e06438207d9f
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Mon Feb 22 20:35:25 2021 -0500

    Reverted all changes except grammar fix

    Reverted Makefile and scanner changes. Will make those changes after
    some scanner fixes are committed.

commit 6ac3b0e965b37430ae50c50993188dee88685f78
Merge: 5624600 49d314d
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 21:08:31 2021 -0500

    WIP on kidus-fix-grammar: 5624600 fixed shift/reduce with reversed parens

commit 49d314daca1344d18d6710b10bd47807a9aabb49
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 21:08:31 2021 -0500
```

```
    index on kidus-fix-grammar: 5624600 fixed shift/reduce with reversed parens

commit 56246001c8c85b7f04f5bfedd2c88e4d72acd66e
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 20:47:13 2021 -0500

    fixed shift/reduce with reversed parens

commit a6d2d2db53c93e5b7ed1c5ec4f4f17a704cbeafa
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 20:42:48 2021 -0500

    index on bruk-shift-reduce: 351bf71 all shift/reduce errors removed except
function calling

commit 351bf711886da72a1ea13b303f62d89c8e06895f
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 20:39:31 2021 -0500

    all shift/reduce errors removed except function calling

commit 146f21e2025cee14de353bc318c79ae07d76088b
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Feb 21 17:13:01 2021 -0500

    Partial fix to shift/reduce conflicts. 48 remaining

commit abf7613c2c70e8819247e7b9349939c9e0943dde
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 17:05:41 2021 -0500

    Change FLOATLIT to float type
```

```
commit 211303c6421e5e6f7437d570451d834537121e77
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 17:01:37 2021 -0500

    Add test char literal to literal test

commit 57fc3490a3cb6e6b174bcf6f69fa29714453fabc
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 16:57:50 2021 -0500

    Add test for literals

commit 8351d1ed5b9ffa33dd16accc652a2971d3377b20
Author: Rediet Bekele <redietbekele@dyn-160-39-255-129.dyn.columbia.edu>
Date:   Sun Feb 21 16:31:20 2021 -0500

    test case for function scanner added

commit 6d42dcbd6d162794b8d4ddd0a9eaf7901cd39636
Author: Rediet Bekele <redietbekele@dyn-160-39-255-129.dyn.columbia.edu>
Date:   Sun Feb 21 16:06:21 2021 -0500

    test case, pretty printing and tokens added

commit 4b6ad0da942c065e07cafbc9e5db60d46c190428
Merge: 0928047 922c1ed
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sun Feb 21 13:36:48 2021 -0500

    Merge pull request #4 from Bruk3/temp
```

```
    Draft implementations of scanner and parser

commit 922c1eda8b13368af1af18e2906fc3d194206db6
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Feb 21 13:16:08 2021 -0500

    Makefile fixes

commit a7ccb73ae65c023dabaf6d62630246ce2cf10fee
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Feb 21 12:55:29 2021 -0500

    Add runtests.sh for testsing workflow

    For each test, there should be two files: the source code (.cnet) file,
    and the expected output.
    1. If the source is expected to fail, it should be named
    "fail-<my-test-1>.cnet" and the expected error output
    should be in a file called "fail-<my-test-1>.err"

    2. If the test is expected to pass,  The name of the test
    file should be "test-<my-test-1>.cnet" and the filename of the expected
    output should be "test-<my-test-1>.out"

commit dda6a0fc752722fcc9e2ce15de4c7789a43b87a3
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 12:30:14 2021 -0500

    Test comments and identifier regex

commit ff28ade99d4d556b3c7638daaf89a87dccc3ada3
Author: KidusAM <km3533@columbia.edu>
```

```
Date:   Sun Feb 21 09:17:12 2021 -0500

    added mod token to scanner

commit f8536831a0419cbca178aea23236e7785a1fd7fa
Merge: 197102a 145bf9a
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 09:07:00 2021 -0500

    pulled from github

commit 197102ab61e7175ac6c859cb0dec7fd20faa6400
Author: KidusAM <km3533@columbia.edu>
Date:   Sun Feb 21 08:55:58 2021 -0500

    removed token not found errors

commit 145bf9a03ae1a6166eb259eb8c2540d6f96fad93
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 02:22:33 2021 -0500

    Revise for testing all tokens w/o regexes

commit d1791680d2e020a1491bc10bdcb8660d3fd7af41
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 01:50:51 2021 -0500

    Revise to test basic delims

commit e44537e758ad59d96c46173cf73248deaad2aab5
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sun Feb 21 01:32:04 2021 -0500
```

```
    Add scanner.mll test capability

    Add print function and narrow token test cases by commenting out other
tokens besides braces

commit 63b8421b8566d626dab814e31b9f17a1cd6d4e06
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Feb 20 20:51:21 2021 -0500

    Add regex for single char

    Missing the regex for chars like 'a' '4' etc.

commit ef49d0a5b4a0211d74c8882f07c162d9f46c6285
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Feb 20 20:41:23 2021 -0500

    Add escape chars part of string regex

    Strings can have escape chars like '\n' in them so add the esc_char to the
str regex

commit 1f4cc7ab2a49873cd8e5e7f19beb92bc37f528b6
Author: Kingsley Neequaye <kingneequaye751@gmail.com>
Date:   Sat Feb 20 19:36:41 2021 -0500

    Fix identifier, int literal, and char literal issues

    normal_id was missing the string for the constructor ID

    int literal had digit* which needs to be digit+
```

```
    CHARLIT was not in the parser

commit b077f406fed889c5fa3c8e176ddc94c446a93c8f
Author: KidusAM <km3533@columbia.edu>
Date:   Sat Feb 20 13:46:21 2021 -0500

    refined grammar rules and addes structmem token

commit fd45550c2c2f0450ee92cbd5268ee2a43485b5a2
Author: KidusAM <km3533@columbia.edu>
Date:   Sat Feb 20 09:48:37 2021 -0500

    added expressions to grammar

commit c2f9cc06e1b7553ef55d2d036d1413bb9a9c16da
Author: KidusAM <km3533@columbia.edu>
Date:   Sat Feb 20 09:37:42 2021 -0500

    added statements to grammar

commit 1f14164066e8f0722d0d98ad8266f0802639efcc
Author: KidusAM <km3533@columbia.edu>
Date:   Sat Feb 20 09:16:26 2021 -0500

    started first draft of grammar

commit c94f8fcf486d6cf9e60e75f3681b697b0eb78bb2
Author: KidusAM <km3533@columbia.edu>
Date:   Sat Feb 20 08:50:40 2021 -0500

    started work on the dot operator in the scanner
```

```
commit 0175f049c90fbf4cbe067e996de37f35ea0c59c2
Author: KidusAM <km3533@columbia.edu>
Date:   Thu Feb 18 23:06:28 2021 -0500

    removed old scanner

commit 09056c7ff77bc8e549c6be8a135f2e391b610563
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 22:43:48 2021 -0500

    updated parser

commit 15fbbd5c53fa3ada508a815f88b3db5210616bf1
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 16:20:30 2021 -0500

    added some of the tokens that need regular expressions

commit 8fb99a851e0e0d721041f05d99118101fe26f589
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 15:44:37 2021 -0500

    added the basic scanner and some minor changes to parser

commit 0928047bfab0244ee892a7caff07340add10f3cb
Merge: d7ace35 c160c23
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Feb 18 23:31:08 2021 -0500

    Merge pull request #3 from Bruk3/temp

    First implementation of scanner
```

```
commit c160c23d54f316bd8dcd8795a522ee6553f51f53
Merge: 0e8ecf8 d7ace35
Author: KidusAM <71148113+KidusAM@users.noreply.github.com>
Date:   Thu Feb 18 23:07:55 2021 -0500

    Merge branch 'main' into temp

commit 0e8ecf862ceac5f1221603abe7a942644c71594b
Author: KidusAM <km3533@columbia.edu>
Date:   Thu Feb 18 23:06:28 2021 -0500

    removed old scanner

commit cbf3fe1e2f7e8fe05aa195e263b12b48330b717f
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 22:43:48 2021 -0500

    updated parser

commit d7ace35c534b0fe34f8dd7d5f2f54647f2cc1c6e
Merge: 5656449 ac4e43b
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Feb 18 17:03:08 2021 -0500

    Merge pull request #1 from Bruk3/bruk-add-skeleton

    Add project skeleton

commit ac4e43b44dd74335c3367539959edda49e1a7d8a
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 16:55:05 2021 -0500
```

```
    Add scanner and start implementing typ and expr rules

commit 692d412848377b27cdae4c49b83d7c8c4c8810b3
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 16:20:30 2021 -0500

    added some of the tokens that need regular expressions

commit c58538e1b984017bf502b0dfbb52f0a4fb2759ff
Author: Kidus Mulu <km3533@columbia.edu>
Date:   Thu Feb 18 15:44:37 2021 -0500

    added the basic scanner and some minor changes to parser

commit c6671db49f6355075f06588ade49f3fa41402e5e
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 07:41:32 2021 -0500

    Fix ci workflow syntax error

commit b1c99b1e19de20094a3473b80b7974f31cf416ca
Merge: c250224 1f4fa76
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 07:35:58 2021 -0500

    Merge branch 'bruk-add-skeleton' of https://github.com/Bruk3/C-net into
bruk-add-skeleton

commit c25022495a868899e9a897a6ac4feb4731afe731
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 07:35:18 2021 -0500
```

```
    Enable slack notification for github workflows

commit 1f4fa764963519f49771d669d5a8cb0bac2dfb4c
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Thu Feb 18 06:36:58 2021 -0500

    Add trailing new line at EOF

commit 423c5846dccb490362e9d318c3202758bdc76a50
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 06:34:13 2021 -0500

    Add github CI workflow

commit eec3d24c2ad01fa090e4c616054e637c5584e230
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Thu Feb 18 06:16:06 2021 -0500

    Add Makefile and parser skeleton

commit 5656449d6d561928738567b472b46e8f8e375ee6
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Feb 16 17:11:18 2021 -0500

    Add instructions for VSCode

commit fdbb696f8b41be73bf4915b831295d8f6b7a835b
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Feb 16 03:19:59 2021 -0500

    Update with vscode plugin recommendation
```

```
commit 8ad24ae4cba1e06cf46f3411a3ae7285e430cc7e
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Tue Feb 16 02:18:10 2021 -0500

    Update README with commit and workflow instructions

commit 1a705e4f600370b7c10150ad63e3c7e84aa65750
Author: Bruk Zewdie <bbz2103@columbia.edu>
Date:   Sun Feb 14 21:41:38 2021 -0500

    Add more instructions to README

commit 497fb0403c6bd0e9a1b6d5e157691cd897819f82
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Feb 13 14:42:51 2021 -0500

    Update README

commit 6e82936535b65434c20c826ab7043e756ba47190
Author: Bruk B Zewdie <32719592+Bruk3@users.noreply.github.com>
Date:   Sat Feb 13 14:25:21 2021 -0500

    Initial commit
```

Source Code

ccnet

```
#!/usr/bin/env python3
```

```
import getopt
import sys
import os
import subprocess
import stat
import shutil

CC          = "gcc"                  # C compiler
LLC         = "llc"                  # LLVM compiler
CNET  = "cnet.native" # CNET compiler
LIBCNET = "libcnet/libcnet-main.a"

USAGEMSG  = """USAGE: {} -[t|a|s|l|b] source """.format(sys.argv[0])

def invalid_options(options, source):
            if len(options) > 2: return True
            if len(source) != 1: return True
            if len(options) == 2 and "o" not in options: return True

            return False




def handle_normal(options, sourcef):
            if 'o' in options: # user specified a file
                        pass

            option = list(options.keys())[0]

            arg1 = "-" + option
            os.execl(CNET, CNET, arg1, sourcef)
            sys.exit(1)

def handle_full(options, sourcef):
            def die(ret):
                        print(ret.stderr.decode(), file=sys.stderr,
end='')
                        sys.exit(ret.returncode)

            # compile cnet to LLVM
            args = ["./" + CNET, "-c", sourcef]
            ret = subprocess.run(args, capture_output=True)
            if (ret.returncode != 0): die(ret)

            # compile LLVM to assembly
            args = [LLC, "-relocation-model=pic"]
            ret = subprocess.run(args, capture_output=True, input=ret.stdout)
            if (ret.returncode != 0): die(ret)

            # compile assembly to C
            tmpfile = "/tmp/" + sourcef.split('/')[-1] + ".s"
            with open(tmpfile, "w") as tmp:
```

```
                                tmp.write(ret.stdout.decode())

            args = [CC, "-g", "-Wall", tmpfile, LIBCNET, "-o", "a.out"]
            ret = subprocess.run(args, capture_output=True)
            if (ret.returncode != 0): die(ret)


            outputf = ""
            if 'o' in options: # user specified a file
                                # with open(options['o'], 'wb') as outfile:
                                #           outfile.write(ret.stdout)
                                shutil.move("a.out", options['o'])
                                os.chmod(options['o'], 0o755)
                                sys.exit(0)
            else:
                                outputf = sourcef.split('.')[0]
                                # with open(outputf, 'wb') as outfile:
                                #           outfile.write(ret.stdout)
                                shutil.move("a.out", outputf)
                                os.chmod(outputf, 0o755) # make it executable
                                sys.exit(0)


def main():
            opts_l, source = getopt.getopt(sys.argv[1:], "taslbco:")
            # print(opts_l, source)

            options = dict()
            for opt,val in opts_l:
                                options[opt[1:]] = val

            # print(options)

            if invalid_options(options, source):
                                print(USAGEMSG)
                                sys.exit(1)


            if "b" not in options and len(options) > 0 and \
            not (len(options) == 1 and 'o' in options): # not a full
compilation
                                handle_normal(options, source[0])
            else:
                                handle_full(options, source[0])


main();
```

cnet.ml

```
module U = Utils;;

type action =Scanner | Ast | LLVM_IR | Sast | Compile

let () =
  let action =  ref Compile in
  let set_action a () = action := a in
  let speclist = [
    ("-t", Arg.Unit (set_action Scanner), "Print the Scanner Tokens");
    ("-a", Arg.Unit (set_action Ast), "Print the AST");
    ("-s", Arg.Unit (set_action Sast), "Print the SAST");
    ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
    ("-c", Arg.Unit (set_action Compile),
      "Check and print the generated LLVM IR (default)");
  ] in
  let usage_msg = "usage: ./cnet.native [-a|-s|-l|-c|-t] [file.cnet]" in
  let channel = ref stdin in
  Arg.parse speclist (fun filename -> channel := open_in filename) usage_msg;
  let lexbuf = Lexing.from_channel !channel in

  try
  (*****************************************************************************
                                    Scanner
   *****************************************************************************)
    let _ = match !action with
        Scanner ->
        let token_string_list =
          let rec next accu =
            match Scanner.tokenize lexbuf with
            | Parser.EOF -> List.rev (Scanner_pp.pretty_print Parser.EOF ::
accu)
            | x   -> next (Scanner_pp.pretty_print x :: accu)
          in next []
        in List.iter (fun x -> print_endline x) token_string_list; exit 0;
      | _ -> () in

  (*****************************************************************************
                                     AST
   *****************************************************************************)
    let ast = Parser.program Scanner.tokenize lexbuf in
    let _ = match !action with
        Ast -> print_string (Ast.string_of_program ast); exit 0
      | _  -> () in

  (*****************************************************************************
                                     SAST
   *****************************************************************************)
    let sast = Semant.check ast in
    let _ = match !action with
      | Sast -> print_string(Sast.string_of_sprogram sast); exit 0
      | _ -> () in

  (*****************************************************************************
                                    Codegen
```

```
   ***************************************************************************)
    let llvm_module = Codegen.translate sast in

    let _ = match !action with
        LLVM_IR ->
        print_string (Llvm.string_of_llmodule llvm_module); exit 0
      | Compile -> Llvm_analysis.assert_valid_module llvm_module;
        print_string (Llvm.string_of_llmodule llvm_module)
      | _ -> ()
    in
    exit 0;

  with
    Parsing.Parse_error ->
    let err_line = U.line_num lexbuf in
    let spec_char = Lexing.lexeme lexbuf in
    let _  = Printf.fprintf stderr "Syntax error on line %d near %s\n"
err_line
        spec_char in
      exit 1

  | Scanner.ScannerError msg ->
    Printf.fprintf stderr "Scanner error: %s\n" msg; exit 1;

  | Sast.SemanticError(msg, _) ->
    Printf.fprintf stderr "Semantic error: %s\n" msg; exit 1;
```

scanner.mll

```
{
open Parser;;
open Utils;;
exception ScannerError of string;;

let scanner_err msg linenum =
  raise (ScannerError (Printf.sprintf msg linenum))
;;

}



let alpha = ['a'-'z' 'A'-'Z']
let digit = ['0'-'9']
let squote = '\''
let bslash = '\\'
let octal_dig = ['0'-'7']
let octal_triplet = (octal_dig)(octal_dig)(octal_dig)
let integer = digit+
```

```
let exp = 'e'['-''+']?['0'-'9']+
let sliterals = ([^ '\\' '"' '\n'] | ('\\'[^ '\n' ]))*
let cfloat = (
    ((digit)+'.'(digit)* (exp)?) |
    ((digit)* '.'(digit)+(exp)?) |
    ((digit)+exp)
)

let normal_id = (alpha | '_')(alpha | digit | '_')*

let whitespace = [' ' '\t' '\r' '\n']

let print_char = [' '-'~']

rule tokenize = parse
[' ' '\t' '\r'] { tokenize lexbuf }
| '\n' { Lexing.new_line lexbuf; tokenize lexbuf }
| '('  { LPAREN }
| ')'  { RPAREN }
| '{'  { LBRACE }
| '}'  { RBRACE }
| '['  { LBRACKET }
| ']'  { RBRACKET }
| ','  { COMMA }
| ';'  { SEMI }
(* Operators *)
| '+' { PLUS }
| '-' { MINUS }
| '*' { TIMES }
| '/' { DIVIDE }
| '=' { ASSIGN }
| '%' { MOD }
| "+=" { PLUSEQ }
| "-=" { MINUSEQ }
| "==" { EQ }
| "!=" { NEQ }
| '<' { LT }
| "<=" { LEQ }
| ">" { GT }
| ">=" { GEQ }
| "&&" { AND }
| "||" { OR }
| "!" { NOT }
| '.' { DOT }
(*Control flow*)
| "if" { IF }
| "else" { ELSE }
| "else" whitespace+ as ws "if" {count_new_lines ws lexbuf; ELIF }
| "for" { FOR }
| "while" { WHILE }
| "break" { BREAK }
| "continue" { CONTINUE }
(*Types*)
| "int" { INT }
```

```
| "float" { FLOAT }
| "char" { CHAR }
| "string" { STRING }
| "void" { VOID }
| "struct" { STRUCT }
| "socket" { SOCKET }
| "file"   { FILE }
(*Functions*)
| "return" { RETURN }
(*Memory*)
| "new" { NEW }
| "delete" { DELETE }

(* Now the tokens that have to be matched with regex *)
| "//" { scomment lexbuf }
| "/*" { mcomment lexbuf }
| normal_id as lxm {ID(lxm)}
| integer as lxm { INTLIT(int_of_string lxm) }
(* | '"' ((print_char)* as str) '"' { STRLIT(str) } *)
| '"' { STRLIT(strlit "" lexbuf) }
| squote bslash ((octal_triplet) as oct_num)  squote { CHARLIT(int_of_string
("0o" ^ oct_num)) }
| squote squote { scanner_err "empty char literal on line %d" (line_num
lexbuf)}
| squote bslash (['f' 'n' 'r' 't' '\\' '0' '\''] as spec_char) squote {
    match spec_char with
      'f' -> CHARLIT(12)
    | 'n' -> CHARLIT(10)
    | 'r' -> CHARLIT(13)
    | 't' -> CHARLIT(9)
    | '\\' -> CHARLIT(92)
    | '0' -> CHARLIT(0)
    | '\'' -> CHARLIT(47)
    | _ -> scanner_err "[COMPILER BUG] unaccounted escape character line %d"
(line_num lexbuf)
}

| squote print_char squote as lxm               {CHARLIT(Char.code(lxm.[1]))}
(*For chars like 'a'*)
| cfloat as flt { FLOATLIT(float_of_string flt) } (* TODO Optional negative
sign *)

(* Error cases *)
| '"' | squote { scanner_err "unmatched quote on line %d" (line_num lexbuf) }
| _ { scanner_err "illegal character on line %d" (line_num lexbuf) }

| eof { EOF }

and strlit str_so_far = parse
    '"' { str_so_far }
| "\\n" { strlit (str_so_far ^ "\n") lexbuf }
| "\\t" { strlit (str_so_far ^ "\t") lexbuf }
| "\\r" { strlit (str_so_far ^ "\r") lexbuf }
| "\\0" { strlit (str_so_far ^ (String.make 1 (Char.chr 0))) lexbuf }
```

```
| "\\\\" { strlit (str_so_far ^ "\\") lexbuf }
| "\""    { strlit (str_so_far ^ "\"") lexbuf }
| "\\" ((octal_triplet) as oct_num)
   { strlit (str_so_far ^ (String.make 1 (Char.chr (int_of_string ("0o" ^
oct_num))))) lexbuf }
| print_char as lxm { strlit (str_so_far ^ (String.make 1 lxm)) lexbuf }
| eof { scanner_err "unmatched quote on line %d" (line_num lexbuf) }
| _ { scanner_err "illegal character in string literal on line %d" (line_num
lexbuf) }

and scomment = parse
'\n' { Lexing.new_line lexbuf; tokenize lexbuf }
| eof { tokenize lexbuf }
| _ { scomment lexbuf }

and mcomment = parse
"*/" { tokenize lexbuf }
| '\n' { Lexing.new_line lexbuf; mcomment lexbuf }
| eof { raise (ScannerError("reached end of file with an unclosed multiline
comment"))}
| _ { mcomment lexbuf }
```

ast.ml

```
(*****************************************************************************
*
                      Abstract syntax tree types for C-net
*****************************************************************************
*)


type unop = Not | Minus
type binop =
      (*  Relational operators  *)
    Eq
  | Neq
  | Lt
  | Leq
  | Gt
  | Geq
      (*  Logical operators  *)
  | And
  | Or
      (*  Arithmetic operators  *)
  | Add
  | Sub
  | Mul
  | Div
  | Mod
```

```
      (* assignment operators *)
type bin_assign_op =
  Assign | PlusEq | MinusEq

      (* 'recursive' id that can be an id or a member of a struct *)
and id = typ * string

and rid =
    FinalID of string
  | RID of rid * string (* my_struct.my_member *)
  | Index of rid * expr (* my_struct.my_member[3] *)

(* So for eg. my_struct.ms2.ms_array[2].my_member is valid *)

                              (* types in C-net *)
and typ =
  Char | Int | Float | String | Socket | File | Struct of string | Void
  | Array of typ


                                (* Expression *)
and newable =
    NStruct of string

and expr =
  Noexpr
  (* Literals *)
  | Intlit of int
  | Charlit of int
  | Floatlit of float
  | Strlit of string
  | Rid of rid
  (* Operators *)
  | Binop of expr * binop * expr
  | Unop of unop * expr
  | Binassop of rid * bin_assign_op * expr
  (* Arrays and new/delete *)
  | New of newable
  | ArrayLit of typ * expr * expr list (* expr:length and expr list:array
literal *)
  (* | Index  of rid * expr *)
  (* Function calls *)
  | Call of rid * expr list


                                (* Statements *)
type vdecl = {vtyp : typ ; vname : string}

type stmt =
    Expr of expr
  | Return            of expr
  | Delete of rid
  | Break
```

```
  | Continue
  | If               of (expr  * stmt) list * stmt
  | For              of expr  * expr * expr * stmt
  | While            of expr  * stmt
  | Vdecl            of vdecl
  | Vdecl_ass      of vdecl * expr
  | Block   of stmt list

                                  (* Functions *)
type func = {t: typ ; name : string ; parameters : id list ; body : stmt list;
locals : id list}

                                   (* Structs *)
type strct = { sname : string ; members : vdecl list }

                                   (* Program *)
type decl =
    GVdecl of vdecl (* Renamed to GVdecl to avoid collision with Vdecl of stmt
which was giving errors*)
  | GVdecl_ass of (vdecl * expr)
  | Sdecl of strct
  | Fdecl of func

type program =
    Program of decl list


                         (* Pretty-printing functions *)
let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mul -> "*"
  | Div -> "/"
  | Eq -> "=="
  | Neq -> "!="
  | Lt -> "<"
  | Leq -> "<="
  | Gt -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"
  | Mod -> "%"

let string_of_uop = function
    Minus -> "-"
  | Not -> "!"

let string_of_binassop = function
    Assign -> "="
  | PlusEq -> "+="
  | MinusEq -> "-="
```

```
(* // TODO  *)


let rec string_of_typ = function
    Char      -> "char"
  | Int       -> "int"
  | Float     -> "float"
  | Socket    -> "socket"
  | File      -> "file"
  | String    -> "string"
  | Struct(t) -> "struct " ^ t
  | Void      -> "void"
  | Array(t)  ->  "" ^ string_of_typ t ^ "[]"

let string_of_id (t, n) = string_of_typ t ^ " " ^ n
let rec string_of_rid = function
  | FinalID(id) -> id
  | RID(r, final) -> string_of_rid r ^ "." ^ final
  | Index(r, expr) -> string_of_rid r ^ "[" ^ (string_of_expr expr) ^ "]"

and string_of_newable = function
    NStruct(n)  -> "struct " ^ n

and string_of_expr = function
  | Noexpr -> ""
  | Intlit(id) -> string_of_int id
  | Charlit(id) -> "" ^ "\'" ^ (Char.escaped(Char.chr(id))) ^ "\'"
  | Floatlit(id) -> string_of_float id
  | Strlit(id) -> "\"" ^ id ^ "\""
  | Rid(id) -> string_of_rid id (* TODO *)
  | Binop(e1, o, e2) ->
    string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Binassop(id, op, r) -> string_of_rid id ^ " " ^ string_of_binassop op ^ "
" ^ string_of_expr r
  | New(n) -> "new " ^  string_of_newable n
  | ArrayLit(t, e, el) ->
    "new " ^ string_of_typ t ^ "[" ^ string_of_expr e ^ "] = {" ^
    String.concat ", " (List.map string_of_expr el) ^ "}"
  | Call(f, el) ->
    string_of_rid f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^
")"


let string_of_vdecl vdecl  =
  string_of_typ vdecl.vtyp ^ " " ^ vdecl.vname ^ ";\n"
let string_of_vdecl_assign (t, id, e) =
  string_of_typ t ^ " " ^ id ^ " = " ^ string_of_expr e ^ ";\n"
let string_of_strct (name, members) =
  "struct " ^ name ^ " {\n\t" ^
  String.concat "\t" (List.map string_of_vdecl members) ^ "};\n\n"

let tabs num = (* tabs 5 returns "\t\t\t\t\t" *)
  let rec helper s =  function
```

```
    | 1 -> "\t" ^ s
    | n when n > 1 -> helper (s ^ "\t") (n-1)
    | _ -> ""
  in helper "" num;;

(* TODO *)
let rec string_of_stmt (main_stmt, main_indent) =
  let print_block b ind = match b with
      Block(_)    -> string_of_stmt (b, ind)
    | _           -> (tabs ind) ^
                     "{\n" ^ string_of_stmt (b, ind + 1) ^
                     (tabs ind) ^ "}\n"
  in
  let helper (stmt, indent) = match stmt with
      Block(stmts)       -> "{\n" ^ String.concat ""
                              (List.map string_of_stmt
                                 (List.map
                                    (fun stmt -> (stmt, indent + 1))
                                    stmts
                                 )
                              )
                            ^ (tabs indent) ^ "}\n"
    | Expr(expr)         -> string_of_expr expr ^ ";\n"
    | Return(expr)       -> "return " ^ string_of_expr expr ^ ";\n"
    | Delete(id)         -> "delete " ^ string_of_rid id ^ ";"
    | Break              -> "break;"
    | Continue           -> "continue;"
    | If(e_s_l,Expr(Noexpr)) -> let string_of_if ((e, s))  =
        "if (" ^ string_of_expr e ^ ")\n"  ^ (print_block s indent)
      in String.concat (tabs indent ^ "else ") (List.map string_of_if e_s_l)
    | If(e_s_l, s)       ->
      let string_of_if ((e, s))  =
        "if (" ^ string_of_expr e ^ ")\n"  ^ (print_block s indent)
      in String.concat (tabs indent ^ "else ") (List.map string_of_if e_s_l) ^
         (tabs indent) ^ "else\n" ^ (print_block s indent)
    | For(e1, e2, e3, s) -> "for (" ^ string_of_expr e1  ^ " ; " ^
                            string_of_expr e2 ^ " ; " ^ string_of_expr e3  ^
") "
                            ^ (print_block s indent)
    | While(e, s)        -> "while (" ^ string_of_expr e ^ ")\n"
                            ^ (print_block s indent)
    | Vdecl(v)           -> string_of_vdecl v
    | Vdecl_ass({vtyp; vname}, e)
      -> string_of_vdecl_assign(vtyp, vname, e)

  in (tabs main_indent) ^ helper (main_stmt, main_indent)


let string_of_func (t, n, p, b) =
  string_of_typ t ^ " " ^ n ^ "(" ^ String.concat "," (List.map string_of_id
p) ^
  ")\n{\n" ^ String.concat ""
    (List.map
       string_of_stmt
```

```
        (List.map (fun stmt -> (stmt, 1)) b)
    ) ^ "}\n\n"

let string_of_decl = function
    GVdecl(vdecl) -> string_of_vdecl vdecl
  | GVdecl_ass({vtyp; vname}, e) -> string_of_vdecl_assign(vtyp, vname, e)
  | Sdecl({sname; members}) -> string_of_strct(sname, members)
  | Fdecl({t; name; parameters; body; _}) -> string_of_func(t, name,
parameters, body)

let string_of_program  = function
  Program(decls) -> String.concat "" (List.map string_of_decl decls) ^ "\n"
```

parser.mly

```
/* Ocamlyacc parser for C-net */

%{
    open Ast
%}

%token LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET COMMA SEMI
%token MOD ASSIGN
%token PLUS MINUS TIMES DIVIDE
%token PLUSEQ MINUSEQ
%token DOT
%token EQ NEQ LT LEQ GT GEQ
%token AND OR NOT
%token IF ELSE ELIF FOR WHILE RETURN BREAK CONTINUE
%token INT FLOAT CHAR STRING VOID STRUCT SOCKET FILE
%token NEW DELETE
%token <int> INTLIT CHARLIT
%token <float> FLOATLIT
%token <string> STRLIT
%token <string> ID
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%left ELIF
%right PLUSEQ MINUSEQ ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%left MOD
%right NOT
```

```
%start program

%type <Ast.program> program




%%

program:
    decls EOF { Program(List.rev $1) }

decls :
    decls decl { $2::$1 }
    | decl { [$1] }

decl:
   | vdecl { GVdecl($1) }
   | vdecl_assign { GVdecl_ass(fst $1, snd $1) }
   | sdecl { Sdecl($1) }
   | fdecl { Fdecl($1) }

typ :
    VOID   { Void }
    | CHAR  { Char }
    | INT  { Int }
    | FLOAT  { Float }
    | STRING  { String }
    | SOCKET  { Socket }
    | FILE   { File }
    | STRUCT ID  { Struct($2) }
    | typ LBRACKET RBRACKET { Array($1) } /* This is kinda awkward */

vdecls:
    vdecls vdecl { $2::$1 }
    | vdecl { [$1] }

vdecl:
    typ ID SEMI { {vtyp = $1; vname = $2} }

sdecl:
        STRUCT ID LBRACE vdecls RBRACE SEMI { {sname = $2; members = List.rev
$4} }

fdecl :
    typ ID LPAREN opt_params RPAREN LBRACE opt_stmts RBRACE
                    {
                        let is_decl = function
                            Vdecl (_) -> true
                            | Vdecl_ass (_) -> true
                            | _ -> false in
                        let local_vars = List.filter is_decl $7 in
                        let to_decl = function
```

```
                            Vdecl({vtyp; vname}) -> (vtyp, vname)
                          | Vdecl_ass ({vtyp; vname}, _) -> (vtyp, vname)
                          | _ -> raise (Failure "Failed to cast an unexpected
stmt type to declaration ")
                       in
                       let locals = List.map to_decl local_vars in

                       {t = $1; name = $2; parameters = $4; body = $7; locals
= locals}
                   }

opt_params :
    { [] }
    | params { List.rev $1 }

params:
    typ ID { [ ($1, $2) ] }
    | params COMMA typ ID { ($3, $4) :: $1 }


opt_stmts:
    { [] }
    | stmts { List.rev $1 }

stmts:
    | stmts stmt { $2::$1 }
    | stmt { [$1] }

vdecl_assign:
    typ ID ASSIGN expr SEMI { ({vtyp = $1; vname = $2}, $4) }

stmt:
    opt_expr SEMI { Expr($1) }
    | RETURN opt_expr SEMI { Return($2) }
    | DELETE id SEMI { Delete($2) }
    | BREAK SEMI  { Break }
    | CONTINUE SEMI { Continue }
    | ifstmt ELSE stmt    { If($1, $3)}
    | ifstmt %prec NOELSE { If($1,Expr(Noexpr))}
    | FOR LPAREN opt_expr SEMI opt_expr SEMI opt_expr RPAREN stmt { For($3,
$5, $7, $9) }
    | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
    | vdecl { Vdecl($1) }
    | vdecl_assign { Vdecl_ass(fst $1, snd $1) }
    | LBRACE stmts RBRACE { Block(List.rev $2) }

ifstmt:
    IF LPAREN expr RPAREN stmt            { [ ($3,$5) ] }
    | ifstmt ELIF LPAREN expr RPAREN stmt { ($4,$6)::$1 }

opt_expr:
    { Noexpr }
    | expr { $1 }
```

```
opt_arraylit:
        { [] }
    | LBRACE args RBRACE { List.rev $2 }

assignment_op:
        ASSIGN { Assign }
        | PLUSEQ { PlusEq }
        | MINUSEQ { MinusEq }
expr:
    INTLIT               { Intlit($1) }
    | CHARLIT            { Charlit($1) }
    | FLOATLIT           { Floatlit($1) }
    | STRLIT             { Strlit($1) }
    | id                 { Rid($1) }
    | LPAREN expr RPAREN  { $2 }
    | expr EQ expr       { Binop($1, Eq, $3) }
    | expr NEQ expr      { Binop($1, Neq, $3) }
    | expr LT expr       { Binop($1, Lt, $3)}
    | expr LEQ expr      { Binop($1, Leq, $3)}
    | expr GT expr       { Binop($1, Gt, $3) }
    | expr GEQ expr      { Binop($1, Geq, $3) }
    | expr AND expr      { Binop($1, And, $3) }
    | expr OR expr       { Binop($1, Or, $3) }
    | expr PLUS expr     { Binop($1, Add, $3) }
    | expr MINUS expr    { Binop($1, Sub, $3) }
    | expr TIMES expr    { Binop($1, Mul, $3) }
    | expr DIVIDE expr   { Binop($1, Div, $3) }
    | expr MOD expr      { Binop($1, Mod, $3) }
    | expr assignment_op expr %prec ASSIGN   {
                                        let f = match $1 with
                                           Rid(rid) -> Binassop(rid, $2, $3)
                                           | _         -> raise
                                           (Parsing.Parse_error)
                                        in f
                                      }
    | MINUS expr %prec NOT { Unop(Minus, $2) }
    | NOT expr { Unop(Not, $2) }
    | NEW STRUCT ID { New(NStruct($3)) }
    | NEW typ LBRACKET expr RBRACKET opt_arraylit { ArrayLit($2, $4, $6) }
    | id LPAREN opt_args RPAREN { Call($1, $3) }

id :
    ID { FinalID($1) }
    | id DOT ID { RID($1, $3) }
    | id LBRACKET expr RBRACKET { Index($1, $3) }

opt_args :
    { [] }
    | args { List.rev $1 }

args :
    expr { [$1] }
    | args COMMA expr { $3 :: $1 }
```

sast.ml

```
(*****************************************************************************
*
                        Semantically checked ast for cnet
*****************************************************************************
*)

open Ast

(* A semantic error in cnet. It contains a message and a line number *)

(* Kidus: The line number is not really implemented and is only there for
 * forward compatibility. Right now the function semant_err takes only a
string,
 * but later it can take a line number as well
 * *)
exception SemanticError of string * int;;

(* The function for raising a semantic error *)
let semant_err (msg : string) =
  raise (SemanticError(msg, -1));;

type sid =
    SFinalID of string
  | SRID of sid * string
  | SIndex of sid * sexpr

and sexpr = typ * sx
and sx =
    SNoexpr
  | SIntlit of int
  | SCharlit of int
  | SFloatlit of float
  | SStrlit of string
  | SId of sid
  (* Operators *)
  | SBinop of sexpr * binop * sexpr
  | SBinassop of sid * bin_assign_op * sexpr
  | SUnop of unop * sexpr
  | SNew of string
  | SArrayLit of typ * sexpr * sexpr list
  | SCall of string * sexpr list

type sstmt =
    SExpr of sexpr
  | SReturn of sexpr
  | SDelete of sexpr
  | SBreak
```

```
  | SContinue
  | SIf of (sexpr * sstmt) list
  | SFor of sexpr * sexpr * sexpr * sstmt
  | SWhile of sexpr * sstmt
  | SVdecl of vdecl
  | SVdecl_ass of vdecl * sexpr
  | SBlock of sstmt list

type sfunc = {
  styp: typ;
  sfname: string;
  sparameters: id list;
  sbody: sstmt list
}

(* here the sid list is typ * name so its the variable declarations *)
(* type sstrct = {sname: string; smembers: sid list} *)

type sdecl =
  | SGVdecl_ass of (vdecl * sexpr)
  | SSdecl of strct
  | SFdecl of sfunc

type sprogram = sdecl list;;

 (* Pretty-printing functions *)

let contains s1 s2 =
  let re = Str.regexp_string s2
  in
      try ignore (Str.search_forward re s1 0); true
      with Not_found -> false
;;

let remove_prefix (str: string) (prefix: string) =
  let strlen = String.length str in
  let prelen = String.length prefix in
  let prefix_found = contains str prefix in
   if prefix_found then String.sub str prelen (strlen - prelen) else str
;;

let rec string_of_sid = function
  | SFinalID(id) -> id
  | SRID(r, final) -> string_of_sid r ^ "." ^ final
  | SIndex(r, sexpr) -> string_of_sid r ^ "[" ^ (string_of_sexpr sexpr) ^ "]"

and string_of_sexpr (t, e) =
    "(" ^ string_of_typ t ^ " : " ^ (match e with
          SNoexpr -> ""
      | SIntlit(l) -> string_of_int l
      | SCharlit(l) -> "" ^ "\'" ^ (Char.escaped(Char.chr(l))) ^ "\'"
      | SFloatlit(l) -> string_of_float l
      | SStrlit(l) -> "\"" ^ l ^ "\""
      | SId(s) -> string_of_sid s
```

```
      | SBinop(e1, o, e2) ->
        string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
    | SBinassop(v, o, e) -> string_of_sid v ^ string_of_binassop o ^
string_of_sexpr e
    | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
    | SNew (n) -> "new " ^ n
    | SArrayLit (t, e, el) ->
    "new " ^ string_of_typ t ^ "[" ^ string_of_sexpr e ^ "] = {" ^
    String.concat ", " (List.map string_of_sexpr el) ^ "}"
    | SCall(f, el) ->
        remove_prefix f "user_" ^ "(" ^ String.concat ", " (List.map
string_of_sexpr el) ^ ")"
          ) ^ ")"
;;

let string_of_svdecl_assign (t, id, e) =
  string_of_typ t ^ " " ^ id ^ " = " ^ string_of_sexpr e ^ ";\n"
;;

let rec string_of_sstmt = function
  SExpr(expr) -> string_of_sexpr expr ^ ";\n";
  | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
  | SDelete(expr) -> "delete " ^ string_of_sexpr expr ^ ";\n";
  | SIf(e_s_l) ->
    let string_of_sif (e, s) =
      "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
    in
    String.concat "else " (List.map string_of_sif e_s_l)
  | SFor(e1, e2, e3, s) ->
      "for (" ^ string_of_sexpr e1  ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
      string_of_sexpr e3  ^ ")\n\t " ^ string_of_sstmt s
  | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt s
  | SVdecl(v)           -> string_of_vdecl v
  | SVdecl_ass({vtyp; vname}, e)
    -> string_of_svdecl_assign(vtyp, vname, e)
  | SBreak -> "break;"
  | SContinue -> "continue;"

  | SBlock(stmts) ->
    "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n";;

let string_of_sfunc (t, n, p, b) =
  string_of_typ t ^ " " ^ remove_prefix n "user_" ^ "(" ^ String.concat ","
(List.map string_of_id p) ^
  ")\n{\n" ^ String.concat ""
    (List.map
       string_of_sstmt
       b
    ) ^ "}\n\n";;

let string_of_sdecl = function
  | SGVdecl_ass({vtyp; vname}, e) -> string_of_svdecl_assign(vtyp, vname, e)
  | SSdecl({sname; members}) -> string_of_strct(sname, members)
  | SFdecl({styp; sfname; sparameters; sbody; _}) -> string_of_sfunc(styp,
```

```
sfname, sparameters, sbody)

    (* TODO *)
let string_of_sprogram (decls : sprogram) =
  String.concat "" (List.map string_of_sdecl decls) ;;
```

semant.ml

```
(* Semantic checking for the MicroC compiler *)

module U = Utils
open Ast
open Sast

module StringMap = Map.Make(String);;

(* Semantic checking of the AST. Returns an SAST if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each function *)

(* Add other things that might be needed inside statements to this struct *)
(* to_free : list of names of string variables to get freed
 * in_loop : specifies whether the current context is a loop
 * *)
type stmt_params = {scp : vdecl StringMap.t list ; fl : string list list; il :
bool};;


let check  = function
    Program(all_decls: decl list) ->
    let is_function = function
        Fdecl(_) -> true
      | _ -> false in
    let func_decl_list = List.filter is_function all_decls in

    let to_ast_func = function
        Fdecl(func) -> func
      | _ -> semant_err "illegal type passed to_ast_func " in

    let functions = List.map  to_ast_func func_decl_list in

    (* Raise an exception if the given rvalue type cannot be assigned to
       the given lvalue type *)
    let check_assign lvaluet rvaluet err =
      if (lvaluet = rvaluet) then lvaluet
      else semant_err err
    in
    let check_assign_builtin lvaluet rvaluet err =
```

```
      if (lvaluet = rvaluet) then
        lvaluet
      else
        match (lvaluet,rvaluet) with
        | (Array(Void), Array(x)) | (Array(x), Array(Void)) -> Array(x)
        | _ -> semant_err err
    in

    let builtin_vars =
    let add_builtinvar m vd = StringMap.add vd.vname vd m in
      List.fold_left add_builtinvar StringMap.empty
        [
          {vname="stdout"; vtyp=File};
          {vname="stdin"; vtyp=File}
        ]
    in

    (* The generic checkbinds function that takes the structs as an argument
     * checks the current variable declaration with all the one's it already
has
     * in the following steps
     * 1) Checks for duplicates within the current scope
     * 2) Checks the validity of the declaration
     *  i) not void
     *  ii) if its a struct, it should be a valid struct
     * If all is well, it returns the the scope updated with the new variable
     * *)
    let rec check_binds_general
        ((scope : vdecl StringMap.t), (structs : strct StringMap.t)) (v :
vdecl)
      : vdecl StringMap.t * strct StringMap.t
      =
      let _ = match StringMap.mem v.vname builtin_vars with
          true -> semant_err (v.vname ^ " cannot be defined as a variable")
        | false -> ()
      in
      let valid_struct (sname : string) = match StringMap.mem sname structs
with
          true -> ()
        | false -> semant_err (v.vname ^ " has unrecognized struct type
[struct " ^ sname ^ "]")
      in

      let _ = match v.vtyp with (* validate non-void / valid struct *)
          Void -> semant_err (v.vname ^ " is a void type, which is illegal")
        | Struct(s) -> valid_struct s
        | Array(t) ->
          ignore (check_binds_general (scope, structs) {vtyp = t; vname =
v.vname ^ "[0]"}); ()
        | _ -> ()
      in

      let _ = match StringMap.mem v.vname scope with (* check no duplicates in
scope *)
```

```
          true -> semant_err ("duplicate " ^ v.vname)
        | false -> ()
      in

      (StringMap.add v.vname v scope) , structs

    in

    (* add the structs of a program to a StringMap and verify that they are
     * valid declarations *)
    let structs : strct StringMap.t =
      let add_struct m = function
          Sdecl(s) ->
          (match StringMap.mem s.sname m with
             true -> semant_err ("Duplicate declaration of struct " ^ s.sname)
           | false ->
             let structs_so_far =
               StringMap.add s.sname s m (* include the current one *)
             in
             ignore (List.fold_left check_binds_general
               (StringMap.empty,structs_so_far) s.members); structs_so_far)
        | _ -> m
      in
      List.fold_left add_struct StringMap.empty all_decls
    in

    (* The specific check binds that already has the structs and takes one
scope
     * (the 'top' one)
     *)
    let check_binds scope v =
      fst (check_binds_general (scope, structs) v)
    in

    (* the check_binds that takes a full scope and checks for conflicts in the
     * top one
     *)
    let check_binds_scoped full_scope v
      : vdecl StringMap.t list
      =
      match full_scope with
        [] -> semant_err ("[COMPILER BUG] empty scope passed to
check_binds_scoped for variable search " ^ v.vname)
      | scope :: tl -> (check_binds scope v) :: tl
    in


    (* Collect global variables and check their validity *)
    let globals =
      let add_global m = function
          GVdecl(vd) | GVdecl_ass(vd, _) -> check_binds m vd
        | Sdecl(_) | Fdecl(_) -> m
      in
      List.fold_left add_global builtin_vars all_decls
```

```
    in
        (* TODO: catch builtin decls *)


(*****************************************************************************
*
                                Built-in functions
*****************************************************************************
*)
    (* @built_in_decls  : this is a string -> fdecl map. However, since there
can
     *                    be multiple built-in functions with different names
     * @builtin_funcs_l : this is a list of all the builtin declarations. This
     *                    is required because the map cannot have two entries
of
     *                    the same name, but some functions such as length are
     *                    defined for multiple objects.
     *)
    let built_in_decls = U.builtin_funcs in
    let builtin_funcs_l = U.builtin_funcs_l in

      (* Add function name to symbol table *)
      let add_func map (fd: func) =
        let built_in_err = "function " ^ fd.name ^ " may not be defined"
        and dup_err = "duplicate function " ^ fd.name
        and n = if fd.name = "main" then fd.name else "user_" ^ fd.name (*
Name of the function prefixed with user_ *)
        in match fd with (* No duplicate functions or redefinitions of
built-ins *)
          _ when StringMap.mem n built_in_decls -> semant_err built_in_err
        | _ when StringMap.mem n map -> semant_err dup_err
        | _ ->  StringMap.add n fd map
      in

      (* Collect all function names into one symbol table *)
      let function_decls = List.fold_left add_func built_in_decls functions
      in

      (* Return a function from our symbol table *)
      let find_func (s : string) =
        try StringMap.find s function_decls
        with Not_found -> semant_err ("unrecognized function " ^ remove_prefix
s "user_")
      in

      (* return a builtin function matched on its name and first parameter *)
      let find_builtin_func name paramt =
        let match_fun = function
            {t=_; name=n; parameters=f :: _; body=_; locals=_} ->
            (if n = name && (fst f = paramt || fst f = Array(Void)) then true
else false)
          | _ -> false
        in
        match List.filter match_fun builtin_funcs_l with
```

```
          hd :: [] -> hd
        | _ -> semant_err ("find_builtin_func asked for " ^ name ^ " that
doesn't exist")
      in

      (* Ensure "main" is defined and has the correct prototype*)
      (* Allowed prototypes for main
        int main()
        int main(string[])
      *)
      let _ = (* check main *)
        let {t=t; name=_ ;body=_ ;locals=_ ; parameters=params} = try
            find_func "main"
          with _ -> semant_err "main function not found"
        in
        let _ = match t with (* check return value *)
            Int -> ()
          | _ -> semant_err  "return type of main must be int."
        in
        match params with
          [] -> ()
        | [(Array(String), _)] -> ()
        | _ -> semant_err "Invalid prototype of main function."
      in

      let check_function func =
        (* Make sure no formals or locals are void or duplicates *)
        (* check_binds "formal" func.formals;
           check_binds "local" func.locals; *)

        (* helper function for finding a variable in either the current scope
or
         * all the scope's that include this one
         *)
        let rec find_var  (vname : string) (scope : vdecl StringMap.t list) :
vdecl = match scope with
            [] -> semant_err ("undeclared identifier " ^ vname)
          | m :: tl ->
            if StringMap.mem vname m then
              let vd = StringMap.find vname m in vd
            else
              find_var vname tl
        in
        (* recursively verify an rid to be valid *)
        let rec type_of_identifier (scope : vdecl StringMap.t list) = function
            FinalID s -> let the_var = find_var s scope in (the_var.vtyp,
SFinalID(s))

          | RID(r, member) ->
            let the_struct, sid = type_of_identifier scope r
            in (match the_struct with
              Struct(sname) ->
              (try
                 let the_struct = StringMap.find sname structs in
```

```
                  match List.filter (fun t -> t.vname = member)
the_struct.members with
                    m :: [] -> m.vtyp, SRID(sid, member)
                  | [] -> semant_err ("struct " ^ sname ^ " has no member " ^
member)
                  | _ -> semant_err ("[COMPILER BUG] struct " ^ sname ^ "
contains multiple members called " ^ member)
                with Not_found -> semant_err ("[COMPILER BUG] variable of type
struct " ^ sname ^
                                              " allowed without the the struct
begin declared"));
              | t -> semant_err ("dot operator not allowed on variable " ^
                                  string_of_rid r ^ " of type " ^ string_of_typ
t))

            | Index(r, e) -> (* TODO: index into a string should be char *)
              let (t, e') = expr scope e in
              match t with
                Int ->
                (let vt, iid = type_of_identifier scope r in
                 match vt with
                   Array(at) -> at, SIndex(iid, (t, e'))
                 | String -> Char, SIndex(iid, (t, e'))
                 | _ -> semant_err ("cannot index non-array variable " ^
                                    (string_of_rid r)))
              | ot -> semant_err ("index into an array has to be of type int, "
^
                    "but the expression (" ^ (string_of_expr e) ^ ") has type
" ^
                    (string_of_typ ot))




          (* in *)
(* Return a semantically-checked expression, i.e., with a type *)

          and expr (scope : vdecl StringMap.t list) = function
              Charlit l -> (Char, SCharlit l)
            | Intlit l -> (Int, SIntlit l)
            | Floatlit l -> (Float, SFloatlit l)
            | Strlit l -> (String, SStrlit l)
            | Noexpr     -> (Void, SNoexpr)
            | Rid rid      ->
              (match rid with
                FinalID(n) when n = "stdin" -> File, SId(SFinalID("cnet_stdin"))
              | FinalID(n) when n = "stdout" -> File,
SId(SFinalID("cnet_stdout"))
              | _ -> let t, id = type_of_identifier scope rid in t, SId(id))
            | Binassop (var, op, e) as ex -> (match op with
                  PlusEq -> expr scope (Binassop(var, Assign, Binop(Rid(var),
Add, e)))
                | MinusEq -> expr scope (Binassop(var, Assign, Binop(Rid(var),
Sub, e)))
```

```
              | Assign -> let lt, lid = type_of_identifier scope var
                and (rt, e') = expr scope e in
                let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
                          string_of_typ rt ^ " in " ^ string_of_expr ex
                in check_assign lt rt err, SBinassop(lid, Assign, (rt, e')))
          | Unop(op, e) as ex ->
            let (t, e') = expr scope e in
            let ty = match op with
                (Minus | Not) when t = Int || t = Float -> t
              | _ -> semant_err ("illegal unary operator " ^
                                 string_of_uop op ^ string_of_typ t ^
                                 " in " ^ string_of_expr ex)
            in (ty, SUnop(op, (t, e')))
          | Binop(e1, op, e2) as e ->
            let (t1, e1') = expr scope e1
            and (t2, e2') = expr scope e2 in
            (* All binary operators require operands of the same type *)
            let same = t1 = t2 in
            (* Determine expression type based on operator and operand types
*)
            let ty = match op with
                Add | Sub | Mul | Div when same && t1 = Int   -> Int
              | Add | Sub | Mul | Div when same && t1 = Float -> Float
              | Add when same && t1 = String -> String
              | Mul when (t1=Int && t2=String) || (t1=String && t2=Int) ->
String
              (* | Add | Sub when t1 = Int && t2 = Char -> Int *)
              (* | Add | Sub when t1 = Char && t2 = Int -> Float *)
              | Eq | Neq            when same               -> Int
              | Lt | Leq | Gt | Geq when same && (t1 = Int || t1 = Float) ->
Int
              | And | Or when same && t1 = Int -> Int
              | _ -> semant_err ("illegal binary operator " ^
                                 string_of_typ t1 ^ " " ^ string_of_op op ^ "
" ^
                                 string_of_typ t2 ^ " in " ^ string_of_expr e)
            in (ty, SBinop((t1, e1'), op, (t2, e2')))
          | Call(fname, args) as call ->

            let fname = (match fname with
                FinalID(f_name) -> FinalID("user_" ^ f_name)
                | _ -> fname

            ) in

            let isbuiltin = StringMap.mem (U.final_id_of_rid fname)
built_in_decls in
            let args = (match fname with
                  FinalID(_) -> args
                | RID(sm,_) when isbuiltin -> Rid(sm) :: args
                | RID(_,r) -> semant_err ("function call on member " ^ r ^ "
is not a builtin function" )
                | indx -> semant_err ("cannot call a function on index " ^
(string_of_rid indx))
```

```
              )
            in

            let fd  = if isbuiltin then
                match args with
                  f :: _  -> find_builtin_func (U.final_id_of_rid fname) (fst
(expr scope f))
                | _ -> semant_err "built-in function called with no arguments"
              else
                (find_func (U.final_id_of_rid fname))

            in

            let check_call fd args =
              let param_length = List.length fd.parameters in
              if List.length args != param_length then
                let expected = if isbuiltin then param_length - 1 else
param_length in
                semant_err ("expecting " ^ string_of_int expected ^ "
arguments in " ^ string_of_expr call)
              else begin
                let check_call_helper (ft, _) e =
                  let (et, e') = expr scope e in
                  let err = "illegal argument found in call to " ^ fd.name ^ "
: " ^ "found " ^
                            string_of_typ et ^ " but expected " ^
string_of_typ ft ^ " in " ^
                            string_of_expr e
                  in ((if isbuiltin then check_assign_builtin else
check_assign) ft et err, e')
                in List.map2 check_call_helper fd.parameters args
              end
            in
            let args' = check_call fd args in
            (fd.t, SCall(U.final_id_of_rid fname, args'))
          | New(NStruct(sn)) ->
              let ty =  try (ignore (StringMap.find sn structs)) ; Struct(sn)
with
                Not_found -> semant_err("invalid new expression: type [struct
" ^ sn ^ "] doesn't exist")
              in (ty, SNew(sn))

          | ArrayLit(t, e, e_l) ->
              let check_int =
                let err = "illegal expression found: new " ^ string_of_typ t ^
"[" ^ string_of_expr e ^
                  "] Expression " ^ string_of_expr e ^ " should be of type int"
                in
                let (t', se) = expr scope e in match t' with
                  Int -> (t', se)
                  | _ -> semant_err(err)
              in
              let check_expr_list e_l =
                let err t1 t2 = "illegal expression found in Array literal.
```

```
Array of type " ^
                  string_of_typ t1 ^ " can not contain element of type " ^
string_of_typ t2
                  in
                let sx_list = List.map (fun e -> expr scope e) e_l in
                let rec invalid_exists = function
                  [] -> sx_list
                  | (ti, _) :: tail -> if ti = t then invalid_exists tail else
semant_err(err t ti)
                  in invalid_exists sx_list
            in
              (Array(t), SArrayLit(t, check_int, (check_expr_list e_l)))
          (* | _ -> semant_err "Expression not yet implemented" *)
        in

        let check_bool_expr scope e =
          let (t', e') = expr scope e
          and err = "expected integer expression in " ^ (string_of_expr e)
          in (if t' != Int then semant_err err else (t', e'))

        in
        let string_flatten exp =
          let pres, e, fres = U.handle_strings exp in match pres with
            [] -> SExpr(e)
          | _ -> SBlock( pres @ [SExpr(e)] @ fres)
        in


          (* Take the current statement and the current scope.  Returns the
new
           * statement and the new scope appropriately.
           * sp : contains the current scope, information about which
variables
           * need to be freed and whether the current context is a loop or not
          *)
        let rec check_stmt (sp : stmt_params) (aexp : stmt)
          : (sstmt * stmt_params)
          =
          let {scp = scope; fl = tofree; il = inloop} = sp in
          let new_scope = {scp = StringMap.empty :: scope ; fl = [] :: tofree;
il = inloop} in
          let new_loop_scope = {scp = StringMap.empty :: scope; fl = [] ::
tofree; il = true} in
          let mkblock = function Block s -> Block s | s -> Block [s] in
          let add_free vd = match tofree with
              [] -> semant_err "[COMPILER BUG] empty list passed to free list"
            | hd :: tl -> match vd.vtyp with
                String -> { scp = check_binds_scoped scope vd;
                            fl = (vd.vname :: hd) :: tl; il = inloop}
              | _ -> {scp = check_binds_scoped scope vd; fl = tofree; il =
inloop}
          in
          let insert_frees {scp=tscp; fl=freelist; il=til} =
            let insert_free vname = SDelete (String, SId(SFinalID(vname))) in
```

```
            match freelist with
              [] -> [], {scp=tscp; fl=[]; il=til}
                    (* semant_err "[COMPILER BUG] empty list passed to
insert_frees"*)
            | hd :: tl -> (List.map insert_free hd), {scp=tscp; fl=tl; il=til}
          in

          match aexp with
            Expr e ->
            let exp = expr scope e in
            string_flatten exp, sp

          | Delete n ->
            let t, id' = type_of_identifier scope n in
            let err = "illegal identifier for delete: [" ^ string_of_typ t ^ "
" ^ string_of_rid n ^
                      "]. Identifier should be of type Struct or Array" in
            let e = Rid(n) in
            let check_valid_delete =  function
                Array(_) | Struct(_) | File | Socket -> SDelete (expr scope
e), sp (* Delete should only be called on arrays and structs *)
              | _ -> semant_err (err)
            in check_valid_delete t

          | Break when inloop -> SBreak, {scp = scope; fl = tofree; il =
false}
          | Break -> semant_err ("break used without being in a loop")
          | Continue when inloop -> SContinue, {scp = scope; fl = tofree; il =
true}
          | Continue -> semant_err ("continue used without being in a loop")

          | If(e_s_l, s) ->
            let sif_of_if (pre_l, fre_l, e_s_l) (e_i,s_i) =
              let e' = check_bool_expr scope e_i in
              let pres, e'', fres = U.handle_strings e' in
              let s_i' = fst (check_stmt new_scope s_i) in
              (pre_l @ pres, fre_l @ fres, (e'' , s_i') :: e_s_l)
            in
            let e_s_l' = (Intlit(1), s) :: e_s_l in
            let (pres, frees, e_s_l'') =
              List.fold_left sif_of_if ([], [], []) e_s_l'
            in
            SBlock(pres @ [SIf(e_s_l'')] @ frees), sp

          | For(e1, e2, e3, st) ->
            let for_blk = Block [Expr(e1); While(e2, Block([st; Expr(e3)]))]
in
            check_stmt sp for_blk
            (* SFor(expr scope e1, check_bool_expr scope e2, expr scope e3,
fst *)
            (*        (check_stmt new_loop_scope (mkblock st))), sp *)

          | While(p, s) ->
            let p' = check_bool_expr scope p in
```

```
            let pres, p'', fres = U.handle_strings p' in
            let s' = match fst (check_stmt new_loop_scope (mkblock s)) with
              SBlock s -> SBlock (s @ (U.strip_decls pres))
              | s -> SBlock ([s] @ (U.strip_decls pres)) (* add the
computations to the end of the while *)
            in
            SBlock(pres @ [SWhile(p'', s')] @ fres), sp

          (* add variable to highest scope *)
          | Vdecl (vd) -> SVdecl_ass(vd, U.default_global vd.vtyp), add_free
vd

          | Vdecl_ass ({vtyp; vname}, e) ->
            let (e', newscp) = expr scope e , check_binds_scoped scope {vtyp;
vname} in
            ignore (expr newscp (Binassop(FinalID(vname), Assign, e)))
            ;
            let pres, e'', fres = U.handle_strings e' in
            let the_decl = SVdecl_ass({vname=vname; vtyp=vtyp},
U.default_global vtyp) in
            let the_ass = SExpr(vtyp, match vtyp with
                String ->
                let the_cp =
                  (String, SCall("cnet_strcpy", [String, SId(SFinalID(vname));
e'']))
                in
                SBinassop(SFinalID(vname), Assign, the_cp)
              | _ -> SBinassop(SFinalID(vname), Assign, e'')

              ) in
            (* let the_assignment = SVdecl_ass({vname=vname; vtyp=vtyp}, e'')
in *)
            SBlock(pres @ [the_decl; the_ass] @ fres) , add_free {vtyp; vname}

          | Return e -> let (t, e') = expr scope e in
            if t = func.t && t != String then SReturn (t, e'), sp
            else if t = func.t && t == String then
              let free_stmts, _ = insert_frees sp in
              SBlock ([ SBlock(free_stmts);
                        SVdecl({vtyp=String; vname="ret_tmp"});
                        string_flatten (String, SBinassop(SFinalID("ret_tmp"),
Assign, (t, e')));
                        SReturn(String, SId(SFinalID("ret_tmp")))
                      ])
            , {scp=scope; fl=[]; il=inloop}
            else semant_err ("return statement in function "^ func.name ^" has
type " ^ string_of_typ t ^
                             " but expected " ^ string_of_typ func.t ^ " in "
^ string_of_expr e)

          (* A block is correct if each statement is correct and nothing
             follows any Return statement.  Nested blocks are flattened. *)
          | Block(sl) ->
            let block_scope = new_scope in
```

```
            let check_stmt_list (* foldable *)
                ((sstmts_so_far : sstmt list), tmp_scope)
                (tmp_stmt : stmt)
              : (sstmt list) * stmt_params =
              let _ = match sstmts_so_far with
                  SReturn(_) :: _ -> semant_err "nothing may follow a return"
                | _ -> ()
              in
              let (sstatement, new_scope) =
                check_stmt tmp_scope tmp_stmt
              in
              (sstatement :: sstmts_so_far, new_scope)
              (* | Block sl :: ss  -> check_stmt_list (sl @ ss) (1* Flatten
blocks *1) *)
              (* | s :: ss         -> fst (check_stmt s block_scope) ::
check_stmt_list ss *)
              (* | []              -> [] *)
            in
            let (checked_block, old_sp) =
              (List.fold_left check_stmt_list ([], block_scope) sl)
            in
            (* (SBlock(List.rev checked_block) *)
            let free_stmts, _ = insert_frees old_sp in
            (match checked_block with
               SReturn(s) :: tl -> SBlock (List.rev (SReturn(s) :: (free_stmts
@ tl)))
             | _ -> SBlock (List.rev (free_stmts @ checked_block))
            ), sp

        in (* body of check_function *)

        { styp = func.t;
          sfname = if func.name = "main" then func.name else "user_" ^
func.name;
          sparameters = func.parameters;
          sbody =
            (* add formals to scope first *)
            let init_params =
              {scp = List.fold_left check_binds_scoped [StringMap.empty;
globals] (U.ids_to_vdecls func.parameters);
               fl = []; il = false;
              }
            in
            match check_stmt init_params (Block(func.body)) with
              (SBlock(sl), _) ->
              (match List.rev sl with (* check there is a return statement for
the function *)
               (* _ -> sl) *)
                 SReturn(_) :: _ when func.t != Void -> sl
               | SBlock(x) :: _ when func.t != Void && (match List.rev x with
                     SReturn(_) :: _ -> true | _ -> false)-> sl
               | _ when func.t = Void -> sl
               | _  -> semant_err ("no return statement found for non-void
function " ^ func.name))
```

```
            | _ -> semant_err "[COMPILER BUG] block didn't become a block?"
        }
      in    (* (globals, List.map check_function functions) *)


      let decl_to_sdecl = function
          GVdecl(vdecl) -> SGVdecl_ass(vdecl, U.default_global vdecl.vtyp)
        | GVdecl_ass(vd, e) -> let t, v = U.compute_global vd e in
          let err = "incompatible type assignment from " ^ string_of_typ
vd.vtyp ^
                    " to " ^ string_of_typ t ^ " in globale variable " ^
vd.vname
          in
          ignore (check_assign t vd.vtyp err); SGVdecl_ass (vd, (t,v))
        (* TODO check assignment*)
        | Sdecl(s) -> SSdecl s
        | Fdecl (func) -> SFdecl (check_function func)

      in
      let sdcls = List.map decl_to_sdecl all_decls in
      sdcls
```

codegen.ml

```
module L = Llvm
module A = Ast
module U = Utils
open Sast
open Ast


module StringMap = Map.Make(String)

(* this should see much fewere uses than SemanticError since codegen should
work
 * relatively perfectly if the semantic checking has passed
 *)
exception CodegenError of string * int;;
(* The function for raising a codegen error *)
let codegen_err (msg : string) =
  raise (CodegenError(msg, -1));;

(* translate : Sast.program -> Llvm.module *)
let translate (sdecl_list : sprogram) =
  (* replace with (vdecls, strct_decls, fdecls) *)
  let (vdecls, sdecls, fdecls) = U.decompose_program sdecl_list in
(* let translate ((vdecls : (A.vdecl * sexpr) list), (strct_decls : strct
list), (fdecls : sfunc list)) = *)
```

```
  let context    = L.global_context () in

  let the_module = L.create_module context "CNet" in

  let find_checked n m =
    if StringMap.mem n m then
      StringMap.find n m
    else
      codegen_err ("[COMPILER BUG] couldn't find id " ^ n)
  in

(*****************************************************************************
**
 *                                    Types

*****************************************************************************
*)

  (* Get types from the context *)
  let i8_t       = L.i8_type     context (* Char *)
  and i32_t      = L.i32_type    context (* Int *)
  and i64_t      = L.i64_type   context  (* Codegen internal use *)
  and i1_t      = L.i1_type   context  (* Codegen internal use *)
  and float_t    = L.double_type context (* Float *)
  and void_t     = L.void_type   context in
  let str_t      = L.pointer_type i8_t in
  let ptr_t t    = L.pointer_type t in


  (* Return the LLVM type for a cnet type *)
  let rec ltype_of_typ (t : A.typ) (cstrcts : (A.strct * L.lltype)
StringMap.t)
    : (L.lltype) = match t with
      A.Char            -> i8_t
    | A.Int             -> i32_t
    | A.Float           -> float_t
    | A.Void            -> void_t
    | A.String          -> ptr_t (snd (find_checked "string" cstrcts))
    | A.Struct(n)       -> ptr_t (snd (find_checked n cstrcts))
    | A.Array(typ)      -> ptr_t (snd (find_checked "array" cstrcts))
    | A.Socket          -> ptr_t (snd (find_checked "cnet_file" cstrcts))
    | A.File            -> ptr_t (snd (find_checked "cnet_file" cstrcts))
  in


  let non_ptr_typ t cstrcts = match t with
    | A.String          ->  (snd (find_checked "string" cstrcts))
    | A.Struct(n)       ->  (snd (find_checked n cstrcts))
    | A.Array(typ)      ->  (snd (find_checked "array" cstrcts))
    | A.Socket          ->  (snd (find_checked "cnet_socket" cstrcts))
    | A.File            ->  (snd (find_checked "cnet_file" cstrcts))
    | _                 ->  ltype_of_typ t cstrcts
  in
```

```
  let  size_of t = match t with
    A.Char            -> 1
  | A.Int             -> 4
  | _                 -> 8
  in

  let type_of t = match t with
    A.Char            -> 0
  | A.Int             -> 0
  | A.Float           -> 1
  | A.String          -> 2
  | _                 -> 3
in

(*****************************************************************************
**
   *                            Declare all the structs

*****************************************************************************
*)
  let cstructs : (A.strct * L.lltype) StringMap.t =
    let declare_struct m (s : strct) =
      let cur_strct = L.named_struct_type context s.sname in
      let m = StringMap.add s.sname (s, cur_strct) m in
      let cmembers =
        Array.of_list (List.map (fun {vname=_; vtyp=t} -> ltype_of_typ t m)
                         s.members)
      in
      let _ = L.struct_set_body cur_strct cmembers false in
      m
    in
    (* TODO: instead of an empty stringmap, the list should be folded on the
     * default struct declarations (io/string/array)
     *)
    let cbuiltinstrcts =
      List.fold_left declare_struct StringMap.empty
        U.builtin_structs_l
    in
    List.fold_left declare_struct cbuiltinstrcts sdecls
  in

  let ltype_of_typ t = ltype_of_typ t cstructs in
  let non_ptr_typ t = non_ptr_typ t cstructs in

  let cbuiltin_vars =
    let declare_struct_var {vtyp=vt; vname=vn} =
      let the_v  = (L.declare_global (ltype_of_typ vt) ("cnet_" ^ vn)
the_module) in
      L.set_externally_initialized true the_v; ({vtyp=vt; vname=vn}, the_v)
    in
    StringMap.fold
      (fun k i m -> StringMap.add ("cnet_" ^ k) (declare_struct_var i) m)
      U.builtin_vars StringMap.empty
```

```
  in


  (* (1* Create a map of global variables after creating each 1 *)
  let global_decls : (A.vdecl * L.llvalue) StringMap.t =
    let global_vdecl m (vd, e) =
      let init = match e with
          A.Float, SFloatlit(f) -> L.const_float (ltype_of_typ A.Float) f
        | A.Int, SIntlit(i)     -> L.const_int (ltype_of_typ A.Int) i
        | A.Char, SCharlit(c)   -> L.const_int (ltype_of_typ A.Char) c
        | A.Struct(n) as t, _   -> L.const_pointer_null (ltype_of_typ t)
        (* | A.String, SStrlit(s)  -> L. *)
        (* | _ -> L.const_int (ltype_of_typ A.Void) 0 *)
      in StringMap.add vd.vname (vd, (L.define_global vd.vname init
the_module)) m in
     List.fold_left global_vdecl cbuiltin_vars vdecls in



(***************************************************************************
**
   *                              Built in functions

****************************************************************************
*)

  (* The function that writes to and reads from sockets/files, including stdin
   * and stdout
  *)
  let builtin_func_decls : (L.llvalue * sfunc) StringMap.t =
    let function_type (fd : sfunc) =
      L.function_type
        (ltype_of_typ fd.styp)
        (Array.of_list (List.map (fun (t,_) -> ltype_of_typ t)
fd.sparameters))
    in
    let declare_func m (fd : sfunc) =
      let the_func = L.declare_function fd.sfname (function_type fd)
the_module in
      StringMap.add fd.sfname (the_func, fd) m
    in

    List.fold_left declare_func StringMap.empty U.sbuiltin_funcs_l
  in


  (* let println_t : L.lltype = *)
  (*   L.function_type i32_t [| i32_t; L.pointer_type i8_t; i32_t |] in *)
  (* let println_func : L.llvalue = *)
  (*   L.declare_function "println" println_t the_module in *)
  let var_arr_t t : L.lltype =
      L.var_arg_function_type (ltype_of_typ (A.Array(t))) [| i32_t; i32_t;
i32_t; i32_t; (ltype_of_typ t) |] in
  let init_array_func t: L.llvalue =
```

```
      L.declare_function "cnet_init_array" (var_arr_t t) the_module in
  let non_variadic_arr_t t : L.lltype =
    L.function_type (ltype_of_typ (A.Array(t))) [| i32_t; i32_t; i32_t |] in
  let cnet_array_decl_func t: L.llvalue =
    L.declare_function "cnet_array_decl" (non_variadic_arr_t t) the_module in
  let arr_idx_t t: L.lltype = match t with
    A.Array(typ) -> L.function_type  (L.pointer_type (ltype_of_typ typ)) [|
ltype_of_typ t; i32_t|]
    | _          -> codegen_err "[COMPILER BUG] Cannot index non-array type"
in
  let cnet_index_arr_func t: L.llvalue =
    L.declare_function "cnet_index_arr" (arr_idx_t t) the_module in

  let cnet_new_str_nolen_t: L.lltype =
    L.function_type (ltype_of_typ A.String) [| ptr_t i8_t |] in
  let cnet_new_str_func: L.llvalue  =
    L.declare_function "cnet_new_str_nolen" cnet_new_str_nolen_t the_module in
  let cnet_empty_str_t: L.lltype =
    L.function_type (ltype_of_typ A.String) [| |] in
  let cnet_empty_str_func: L.llvalue  =
    L.declare_function "cnet_empty_str" cnet_empty_str_t the_module in
  let cnet_char_at_t: L.lltype =
    L.function_type (ltype_of_typ A.Char) [|(ltype_of_typ A.String); i32_t |]
in
  let cnet_char_at_func: L.llvalue =
    L.declare_function "cnet_char_at" cnet_char_at_t the_module in

  let memset_t =
    L.function_type str_t [|str_t; i32_t; i64_t |] in
  let memset_func =
    L.declare_function "memset" memset_t the_module in

  let main_t : L.lltype =
      L.function_type (ltype_of_typ Int) [| i32_t; ptr_t (ptr_t i8_t)|] in
  let main_func = L.declare_function "main" (main_t) the_module in


(************************************************************************
**
   *                            Function signatures

************************************************************************
*)

  (* Define each function (arguments and return type) so we can
     call it even before we've created its body *)
  let function_decls : (L.llvalue * sfunc) StringMap.t =
    let function_decl m (fdecl : sfunc) =
      let ftyp = fdecl.styp
      and name = fdecl.sfname
      and formal_types =
        Array.of_list (List.map
                         (function (t,_) -> ltype_of_typ t)
                         fdecl.sparameters) in
```

```
      let ftype = L.function_type (ltype_of_typ ftyp) formal_types in
      StringMap.add name (L.define_function name ftype the_module, fdecl) m in
    List.fold_left function_decl builtin_func_decls fdecls in

  let find_func_full fname =
    if StringMap.mem fname function_decls then
      StringMap.find fname function_decls
    else
      codegen_err ("[COMPILER BUG] couldn't find function" ^ fname)
  in
  let find_func fname = fst (find_func_full fname) in


(************************************************************************
**
   *                               Function bodies

************************************************************************
*)

  (* Fill in the body of the given function *)
  let build_function_body (fdecl : sfunc) =
    let (the_function, _) =
      if StringMap.mem fdecl.sfname function_decls then
        StringMap.find fdecl.sfname function_decls
      else
        codegen_err ("[COMPILER BUG] couldn't find function" ^ fdecl.sfname)

    in
    let builder = L.builder_at_end context (L.entry_block the_function) in

    (* Construct the function's "locals": formal arguments and locally
       declared variables.  Allocate each on the stack, initialize their
       value, if appropriate, and remember their values in the "locals" map *)
      let local_vars : (A.vdecl * L.llvalue) StringMap.t=
        let add_formal m (t, n) p = L.set_value_name n p;
          let local = L.build_alloca (ltype_of_typ t) n builder in
                ignore (L.build_store p local builder);
          StringMap.add n ({vtyp=t;vname=n},local) m

        and add_local m (t, n) =
          let local_var = L.build_alloca (ltype_of_typ t) n builder
          in StringMap.add n ({vtyp=t;vname=n}, local_var) m

      in
      List.fold_left2 add_formal StringMap.empty fdecl.sparameters
            (Array.to_list (L.params the_function))
        (* in *)
      (* List.fold_left add_local formals fdecl.sparameters *)
      in
      let func_scope = [local_vars; global_decls]
      in
      (* Return the value for a variable or formal argument.
        Check local names first, then global names *)
```

```
      let rec lookup_helper (n : string) scope : (A.vdecl * L.llvalue) =
        match scope with
          [] -> codegen_err ("[COMPILER BUG] cannot find variable " ^ n)
        | hd :: tl ->
          if StringMap.mem n hd then
            StringMap.find n hd
          else
            lookup_helper n tl
      in

      (* Todo: Recursive lookup for complex data types*)
      (* let lookup n scopes = lookup_helper n (lookup_scope n scopes) *)
      (* in *)

      (* Construct code for an expression; return its value *)

      let rec lookup n scope builder = match n with
          SFinalID s -> lookup_helper s scope;
        | SRID(r, member) ->
          let vd, ll = lookup r scope builder in
          let sname = match vd.vtyp with Struct(n) -> n in
          let sd,s = find_checked sname cstructs in
          let the_struct = L.build_load ll "tmp" builder in
          (vd, L.build_struct_gep the_struct (U.mem_to_idx sd member) ""
builder)
        | SIndex(r, ex) ->
          let vd, arr = lookup r scope builder in
          let ll_arr = L.build_load arr "arr" builder in
              match vd.vtyp with
              String -> vd, L.build_call cnet_char_at_func [| ll_arr; expr
builder ex scope |] "charAt" builder;
            | _       -> vd, L.build_call (cnet_index_arr_func (vd.vtyp)) [|
                ll_arr; expr builder ex scope |] "idx_elt" builder

      and expr builder ((t, e) : sexpr) scope  =
        let lookup n = lookup n scope builder in

        match e with
          SNoexpr     -> L.const_int i32_t 0
        | SIntlit i   -> L.const_int i32_t i
        | SCharlit c  -> L.const_int i8_t c
        | SFloatlit f -> L.const_float float_t f
        | SId s       -> L.build_load (snd (lookup s)) (U.final_id_of_sid s)
builder

        | SBinassop (s, op, e) -> let e' =  expr builder e scope
          in (match e with
                _, SNoexpr ->
                let the_null = (L.build_sext_or_bitcast e' (ltype_of_typ t)
"tmp" builder) in
                ignore (L.build_store the_null (snd (lookup s)) builder); e'
              | _ ->
                ignore(L.build_store e' (snd (lookup s)) builder); e'
            )
```

```
        | SBinop ((A.Float,_ ) as e1, op, e2) ->
          let e1' = expr builder e1 scope
          and e2' = expr builder e2 scope in
          (match op with
             A.Add     -> L.build_fadd
           | A.Sub     -> L.build_fsub
           | A.Mul     -> L.build_fmul
           | A.Div     -> L.build_fdiv
           | A.Mod     -> L.build_frem
           | A.Eq      -> L.build_fcmp L.Fcmp.Oeq
           | A.Neq     -> L.build_fcmp L.Fcmp.One
           | A.Lt      -> L.build_fcmp L.Fcmp.Olt
           | A.Leq     -> L.build_fcmp L.Fcmp.Ole
           | A.Gt      -> L.build_fcmp L.Fcmp.Ogt
           | A.Geq     -> L.build_fcmp L.Fcmp.Oge
          | A.And | A.Or ->
              codegen_err "internal error: semant should have rejected and/or
on float"
          ) e1' e2' "tmp" builder
      | SBinop (e1, op, e2) ->
        let result =
                let e1' = expr builder e1 scope
                and e2' = expr builder e2 scope in
                (match op with
                    A.Add       -> L.build_add
                    | A.Sub     -> L.build_sub
                    | A.Mul     -> L.build_mul
                    | A.Div     -> L.build_sdiv
                    | A.Mod     -> L.build_srem
                    | A.And     -> L.build_and
                    | A.Or      -> L.build_or
                    | A.Eq      -> L.build_icmp L.Icmp.Eq
                    | A.Neq     -> L.build_icmp L.Icmp.Ne
                    | A.Lt      -> L.build_icmp L.Icmp.Slt
                    | A.Leq     -> L.build_icmp L.Icmp.Sle
                    | A.Gt      -> L.build_icmp L.Icmp.Sgt
                    | A.Geq     -> L.build_icmp L.Icmp.Sge
                ) e1' e2' "tmp" builder
        in
        L.build_sext_or_bitcast result i32_t "tmp_cast" builder
      | SUnop (op,  ((t, _) as e)) -> let e' = expr builder e scope in
                                            (match op with
                                               A.Minus when t = A.Float ->
L.build_fneg
                                             | A.Minus                  ->
L.build_neg
                                             | A.Not                  ->
L.build_not) e' "tmp" builder
      | SStrlit s   ->
        L.build_call cnet_new_str_func [| L.build_global_stringptr s "tmp"
                                          builder |] "strlit" builder
      | SNew s      ->
        let strct, ll_strct = StringMap.find s cstructs in
        let new_struct = L.build_malloc ll_strct "tmp" builder in
```

```
        let str_members = List.filter (fun mem -> mem.vtyp = A.String )
strct.members in
        let empty_str = fun s -> L.build_call cnet_empty_str_func [| |]
"empty_str" builder in
        let smem_ptr mem = L.build_struct_gep new_struct (U.mem_to_idx strct
mem.vname) "tmp" builder in
        let build_empty_str = fun smem -> ignore(L.build_store (empty_str
smem) (smem_ptr smem) builder) in
        List.iter build_empty_str str_members; new_struct

      | SArrayLit (t, s, arr_lit) ->
        let size_t = expr builder (A.Int,SIntlit((size_of t))) scope in
        let arr_len = expr builder s scope in
        let arr_lit_len = expr builder (A.Int,SIntlit((List.length arr_lit)))
scope in
        let type_t = expr builder (A.Int,SIntlit((type_of t))) scope in
        let ll_arr_lit = List.map (fun a -> expr builder a scope) arr_lit in
        let ll_va_args =  size_t :: type_t :: arr_len ::arr_lit_len ::
ll_arr_lit in
        if ((List.length ll_arr_lit) = 0)
        then L.build_call (cnet_array_decl_func t) [| size_t ; type_t ;
arr_len |] "cnet_array_decl" builder
        else L.build_call (init_array_func t) (Array.of_list ll_va_args)
"cnet_init_array" builder
      | SCall (n, args) ->
        let (fdef, fdecl) = find_checked n function_decls in
        let llargs = List.map (fun a -> expr builder a scope) args in
        let result = (match fdecl.styp with
                            A.Void -> ""
                          | _ -> n ^ "_result") in
              L.build_call fdef (Array.of_list llargs) result builder in
        let add_terminal builder instr =
          match L.block_terminator (L.insertion_block builder) with
            Some _ -> ()
          | None -> ignore (instr builder)




       (* LLVM insists each basic block end with exactly one "terminator"
          instruction that transfers control.  This function runs "instr
builder"
          if the current block does not already have a terminator.  Used,
          e.g., to handle the "fall off the end of the function" case. *)

    in


    (* Build the code for the given statement; return the builder for
       the statement's successor (i.e., the next instruction will be built
       after the one generated by this call) *)

    (* LLVM insists each basic block end with exactly one "terminator"
         instruction that transfers control.  This function runs "instr
```

```
builder"
         if the current block does not already have a terminator.  Used,
         e.g., to handle the "fall off the end of the function" case. *)
      let add_terminal builder instr =
        match L.block_terminator (L.insertion_block builder) with
          Some _ -> ()
        | None -> ignore (instr builder) in

    let add_var (vd, ll) scope = match scope with
        [] -> codegen_err "[COMPILER BUG] empty scope too far into pipeline"
      | hd :: tl -> (StringMap.add vd.vname (vd, ll) hd) :: tl
    in

    (* stmt : ((vdecl * L.llvalue StringMap.t), L.builder) -> sstmt ->
     *          vdecl * L.llvalue StringMap.t
     *)
    let rec stmt (scope,builder) the_stmt =
      let new_scope = StringMap.empty :: scope in

      match the_stmt with
        SExpr e -> ignore(expr builder e scope); (scope, builder)

      | SVdecl vd ->
        stmt (scope,builder) (SVdecl_ass(vd, U.default_global vd.vtyp))
        (* let new_var = *)
        (*   L.build_alloca (ltype_of_typ vd.vtyp) vd.vname builder *)
        (* in *)
        (* (add_var (vd,new_var) scope), builder *)

      | SVdecl_ass (vd, (t, e)) ->
        let new_var = L.build_alloca (ltype_of_typ vd.vtyp) vd.vname builder
        in
        let new_scope = add_var (vd,new_var) scope in
        let the_assignment = vd.vtyp, SBinassop(SFinalID(vd.vname), Assign,
(t,e)) in
        ignore (expr builder the_assignment new_scope); (* do the assignment
*)
        (new_scope, builder)

      | SDelete e ->let t, e' = e in (match t with
          Struct(n) ->(match e' with
                       SId s ->
                         (* Printf.fprintf stderr "To be del:%s\n"
(U.final_id_of_sid s); *)
                         let sd, ll =  lookup s scope builder in
                         let to_be_deleted = L.build_load ll
(U.final_id_of_sid s) builder in
                         let sname = match sd.vtyp with Struct(n) -> n in
                         let sd, _ = find_checked sname cstructs in
                         let delete_str = fun s -> L.build_call (find_func

"cnet_free")
                             [| L.build_sext_or_bitcast s (ltype_of_typ
A.File)
```

```
                                "cast" builder |] "tmp" builder in
                            let smem_ptr mem = L.build_struct_gep to_be_deleted
(U.mem_to_idx sd mem.vname) "tmp" builder in
                            let del_func = fun smem -> ignore(delete_str
(L.build_load (smem_ptr smem) "tmp" builder)) in
                            let str_members = List.filter (fun mem -> mem.vtyp =
A.String ) sd.members in
                            List.iter del_func str_members;
                            L.build_free to_be_deleted builder)

          | _ -> let to_be_deleted = expr builder e scope in
            let to_be_deleted = L.build_sext_or_bitcast to_be_deleted
                (ltype_of_typ A.File) "cast" builder in
                 L.build_call (find_func "cnet_free") [| to_be_deleted |]
"tmp" builder
      );(scope, builder)

      | SReturn e -> ignore (match fdecl.styp with
          (* Special "return nothing" instr *)
            A.Void -> L.build_ret_void builder
          (* Build return statement *)
          | _ -> L.build_ret (expr builder e scope) builder)
                 ; scope, builder

      (* do not attempt *)
      | SIf (psl) ->
        (* let if_elif_bb = L.append_block context "if_elif" the_function in
*)
        let add_if (if_bbs, my_pred_bb) (if_pred, then_stmt) =
          (* cast the value to a bool (1 bit) *)
          let mpbb_builder = L.builder_at_end context my_pred_bb in
          let pred_val = expr mpbb_builder if_pred scope  in
          let pred_val =
            L.build_icmp L.Icmp.Ne pred_val (L.const_int i32_t 0) "tmp"
mpbb_builder
          in
          let my_then_bb = L.append_block context "if_body" the_function in
          let next_bb = L.append_block context "elif" the_function in
          ignore (L.build_cond_br pred_val my_then_bb next_bb mpbb_builder);
          ignore (stmt (new_scope, L.builder_at_end context my_then_bb)
then_stmt);
          (my_then_bb :: if_bbs, next_bb)
        in

        (* we'll start it off in the 'main' bb *)
        let first_bb = L.insertion_block builder in
        let if_bbs, else_pred_bb  = List.fold_left add_if ([], first_bb) psl
in
        (* let else_then_bb = L.append_block context "else_then" the_function
in *)

        (* we don't need the last bb *)
        let _ = L.delete_block else_pred_bb in
```

```
        (* If all else fails, go to the else case *)
        (* let _ = L.build_br else_then_bb (L.builder_at_end context
else_pred_bb) in *)
        (* let _ = (stmt (new_scope, L.builder_at_end context else_then_bb)
else_stmt) in *)

        let merge_bb = L.append_block context "if_merge" the_function in
        let _ = List.map
            (fun bb -> add_terminal (L.builder_at_end context bb) (L.build_br
merge_bb))
            if_bbs

        in
        scope, L.builder_at_end context merge_bb


      (*   let predicate_list = List.map (fun (p,_) -> expr builder p scope)
psl in *)
      (*   let merge_bb = L.append_block context "merge" the_function in *)
      (*   let build_br_merge = L.build_br merge_bb in (1* partial function
*1) *)

      (*   let add_ifthen (predicate, then_stmt) = *)
      (*     let pred = expr builder predicate scope in *)
      (*     let then_bb = L.append_block context "then" the_function in *)
      (*     (add_terminal (snd (stmt (scope, (L.builder_at_end context
then_bb)) then_stmt)) build_br_merge)  :: l *)
      (*   in *)


      (*   let _ = List.fold_left add_ifthen [] psl in *)

      (*   let else_bb = L.append_block context "else" the_function in *)
      (*   add_terminal (stmt (L.builder_at_end context else_bb) else_stmt) *)
      (*     build_br_merge *)


      | SWhile (pred, body) ->
        let pred_bb = L.append_block context "while" the_function in
        ignore (L.build_br pred_bb builder);

        let body_bb = L.append_block context "while_body" the_function in
        let builder' = snd (stmt (new_scope, (L.builder_at_end context
body_bb)) body) in
        add_terminal builder' (L.build_br pred_bb);

        let pred_builder = L.builder_at_end context pred_bb in
        let pred_val = expr pred_builder pred scope  in
        (* cast the value to a bool (1 bit) *)
        let pred_val = L.build_icmp L.Icmp.Ne pred_val (L.const_int i32_t 0)
            "tmp" pred_builder in
```

```
        let merge_bb = L.append_block context "merge" the_function in
        ignore (L.build_cond_br pred_val body_bb merge_bb pred_builder);
        (scope, L.builder_at_end context merge_bb)

      | SBlock(sl) ->
        List.fold_left stmt (new_scope, builder) sl


      | _ -> codegen_err "unimplemented statement type"
    in
    List.fold_left stmt (func_scope, builder) fdecl.sbody; () in
  List.iter build_function_body fdecls; the_module;
```

scanner_pp.ml

```
open Parser;;

let pretty_print = function
  | LPAREN                -> Printf.sprintf "LPAREN"
  | RPAREN                -> Printf.sprintf "RPAREN"
  | LBRACE                -> Printf.sprintf "LBRACE"
  | RBRACE                -> Printf.sprintf "RBRACE"
  | RBRACKET              -> Printf.sprintf "RBRACKET"
  | LBRACKET              -> Printf.sprintf "LBRACKET"
  | EOF                   -> Printf.sprintf "EOF"
  | COMMA                 -> Printf.sprintf "COMMA"
  | SEMI                  -> Printf.sprintf "SEMI"
  | PLUS                  -> Printf.sprintf "PLUS"
  | MINUS                 -> Printf.sprintf "MINUS"
  | TIMES                 -> Printf.sprintf "TIMES"
  | DIVIDE                -> Printf.sprintf "DIVIDE"
  | ASSIGN                -> Printf.sprintf "ASSIGN"
  | PLUSEQ                -> Printf.sprintf "PLUSEQ"
  | MINUSEQ               -> Printf.sprintf "MINUSEQ"
  | EQ                    -> Printf.sprintf "EQ"
  | NEQ                   -> Printf.sprintf "NEQ"
  | NOT                   -> Printf.sprintf "NOT"
  | LT                    -> Printf.sprintf "LT"
  | LEQ                   -> Printf.sprintf "LEQ"
  | GT                    -> Printf.sprintf "GT"
  | GEQ                   -> Printf.sprintf "GEQ"
  | AND                   -> Printf.sprintf "AND"
  | OR                    -> Printf.sprintf "OR"
  | DOT                   -> Printf.sprintf "DOT"
  | MOD                   -> Printf.sprintf "MOD"
  | IF                    -> Printf.sprintf "IF"
  | ELSE                  -> Printf.sprintf "ELSE"
  | ELIF                  -> Printf.sprintf "ELIF"
  | FOR                   -> Printf.sprintf "FOR"
```

```
  | WHILE                -> Printf.sprintf "WHILE"
  | BREAK                -> Printf.sprintf "BREAK"
  | CONTINUE             -> Printf.sprintf "CONTINUE"
  | INT                  -> Printf.sprintf "INT"
  | FLOAT                -> Printf.sprintf "FLOAT"
  | CHAR                 -> Printf.sprintf "CHAR"
  | STRING               -> Printf.sprintf "STRING"
  | SOCKET               -> Printf.sprintf "SOCKET"
  | FILE                 -> Printf.sprintf "FILE"
  | STRUCT               -> Printf.sprintf "STRUCT"
  | VOID                 -> Printf.sprintf "VOID"
  | RETURN               -> Printf.sprintf "RETURN"
  | NEW                  -> Printf.sprintf "NEW"
  | DELETE               -> Printf.sprintf "DELETE"
  | ID(x)                -> Printf.sprintf  "ID(%s)" (x)
  | INTLIT(x)            -> Printf.sprintf  "INTLIT(%d)" (x)
  | STRLIT(x)            -> Printf.sprintf  "STRLIT(%s)" (x)
  | CHARLIT(x)           -> Printf.sprintf  "CHARLIT(%d)" (x)
  | FLOATLIT(x)          -> Printf.sprintf  "FLOATLIT(%f)" (x)
```

Makefile

```
LIB_DIR = ./libcnet
CTEST_DIR = ./tests/stdlib

############################## TEST TARGETS
####################################
test: ccnet
      ./runtests.sh
      @echo "SUCCESS"

############################## cnet top level
#################################

## cnet top level compiler that compiles and links a cnet source file into an
executable
.PHONY: ccnet
ccnet: nobin cnet.native stdlib
      cd $(LIB_DIR) && make


# cnet compiler that translates .cnet -> .ll
# supports the following flags:
# -t : scanner pretty printing
# -a : ast pretty printing
# -s : sast pretty printing
# -l : llvm IR pretty printing (no llvm checking)
# -c : compiler (with llvm checking)
cnet.native: cnet
```

```
	opam config exec -- \
		ocamlbuild -use-ocamlfind cnet.native

######################## Dependencies for ocamlbuild
############################

.PHONY: scanner parser ast

cnet: cnet.ml scanner parser ast scanner_pp

scanner: scanner.mll parser.mly utils.ml
scanner_pp: scanner_pp.ml
parser: parser.mly ast.ml
ast: ast.ml

######################## Std Lib ############################
.PHONY: stdlib stdlib_tests

stdlib_tests:
	cd $(CTEST_DIR) && make all > /dev/null && make clean

stdlib:
	cd $(LIB_DIR) && make all > /dev/null

############################  Other targets
################################
.PHONY: clean
clean:
	opam config exec -- ocamlbuild -clean
	rm -rf final parser.ml parser.mli scanner.ml parser.output \
	scanner.ml scannertest scannertest.out *cmi *cmo \
	*.log *.diff *.out *.err *.ll *.s *.o parser.output \
	tests/integration/*.exe a.out a.out.dsYM/
	cd $(LIB_DIR) && make clean
	cd $(CTEST_DIR) && make clean

.PHONY: all
all: clean ccnet

# to make ocamlbuild happy
.PHONY: nobin
nobin:
	rm -f *.o *.a libcnet/*.o libcnet/*.a
```

utils.ml

```
module A = Ast
open Sast;;
open Ast;;
```

```
module StringMap = Map.Make(String);;

                              (* Scanner utils *)
let count_new_lines whitespace lexbuf =
  String.iter
    (fun c -> if c = '\n' then Lexing.new_line lexbuf else ()) whitespace;;

(* Get the current line number from the lexbuf *)
let line_num lexbuf = lexbuf.Lexing.lex_curr_p.Lexing.pos_lnum


                                (* SAST utils *)
(* Gets the FinalID part of a recursive id. For example, it extracts println
 * from my_struct.my_other_struct.my_file.println
 *)
let rec final_id_of_rid = function
  A.FinalID(fid) -> fid
  | A.RID(_, mem) -> mem
  | A.Index(rid, _) -> final_id_of_rid rid
;;

let rec final_id_of_sid = function
  SFinalID(fid) -> fid
  | SRID(_, mem) -> mem
  | SIndex(sid, _) -> final_id_of_sid sid
;;

(* Get a default value for a global variable based on its type *)
let default_global = function
    A.Char -> (A.Char, SCharlit(0))
  | A.Int -> (A.Int, SIntlit(0))
  | A.Float -> (A.Float, SFloatlit(0.0))
  | A.String -> (A.String, SStrlit(""))
  | A.Void   -> semant_err "[COMPILER BUG] uncaught void global variable
detected"
  | t -> (t, SNoexpr)
;;

(* compute the value of a global variable assignment at compile time. Global
 * variables can only be assigned to constant expressions at declaration *)
let compute_global vdecl exp =
  let verify_types (t1 : A.typ) (t2 : A.typ) =
    if t1 = t2 then ()
     else (semant_err ("incompatible types " ^ A.string_of_typ t1 ^ " and " ^
           A.string_of_typ t2 ^ " in global variable " ^ vdecl.A.vname))
  in
  let bool_int b = if b then 1 else 0 in

  let rec eval_constant = function
      A.Noexpr | A.Binassop(_) | A.New(_) | A.ArrayLit(_) | A.Call(_) ->
      semant_err ("non-constant expression used for global variable " ^
vdecl.A.vname)
    | A.Rid(_) ->
```

```
      semant_err "global declaration using another global variable not
implemented"

    | A.Intlit(i) -> (A.Int, SIntlit(i))
    | A.Charlit(c) -> (A.Int, SIntlit(c))
    | A.Floatlit(f) -> (A.Float, SFloatlit(f))
    | A.Strlit(s) -> (A.String, SStrlit(s))
    | A.Binop(e1, op, e2) -> let e1' = eval_constant e1 and e2' =
eval_constant e2
      in verify_types (fst e1') (fst e2');
      (match (e1', e2') with
         ((A.Int, SIntlit(i)), (A.Int, SIntlit(i2))) ->
         (match op with
            A.Add -> (A.Int, SIntlit(i + i2))
          | A.Sub -> (A.Int, SIntlit(i - i2))
          | A.Mul -> (A.Int, SIntlit(i * i2))
          | A.Div -> (A.Int, SIntlit(i / i2))
          | A.Mod -> (A.Int, SIntlit(i mod i2))
          | A.And -> (A.Int, SIntlit(bool_int ((i land i2) != 0)))
          | A.Or  -> (A.Int, SIntlit(bool_int ((i lor i2) != 0))) (* TODO:
these what we want? *)
          | A.Eq  -> (A.Int, SIntlit(bool_int (i = i2)))
          | A.Neq -> (A.Int, SIntlit(bool_int (i != i2)))
          | A.Lt  -> (A.Int, SIntlit(bool_int (i < i2)))
          | A.Gt  -> (A.Int, SIntlit(bool_int (i > i2)))
          | A.Geq -> (A.Int, SIntlit(bool_int (i >= i2)))
          | A.Leq -> (A.Int, SIntlit(bool_int (i <= i2)))
         )
         (* TODO *)
       | (A.Float, SFloatlit(_)), (A.Float, SFloatlit(_))
       | (A.Float, SFloatlit(_)), (A.Int, SIntlit(_)) (*float string
operations *)
       | (A.Int, SIntlit(_)), (A.Float, SFloatlit(_)) (*float string
operations *)
       | (A.Int, SIntlit(_)), (A.String, SStrlit(_)) (* string int operations
*)
       | (A.String, SStrlit(_)), (A.Int, SIntlit(_)) (* string int operations
*)
                                    -> semant_err "global expression type not
implemented"
       | _ -> semant_err ("non-constant expression used for global variable "
^ vdecl.A.vname)
      )
    | A.Unop(op, e) ->
      let _, e' = eval_constant e in
      match e' with
        SIntlit(i) -> ( match op with
            A.Not -> (A.Int, SIntlit(bool_int (i == 0)))
          | A.Minus -> (A.Int, SIntlit( -1 * i ))
        )
      | _ -> semant_err ("operator [" ^ (A.string_of_uop op) ^ "] only valid
for integers")

  in
```

```
  eval_constant exp ;;

let ids_to_vdecls (ids : A.id list)=
  let id_to_vdecl ((t, n) : A.id) =
    {A.vtyp = t; A.vname = n}
  in List.map id_to_vdecl ids;;

(* Goes through an expression and substitues operations on strings with the
call
 * to the appropriate string library functions. It also creats temporary
 * variables for handling return values of things
 * *)
let handle_strings sexp : sstmt list * sexpr * sstmt list=
  let assign a b = SVdecl_ass({A.vtyp=String; A.vname = a}, b) in
  let styp = A.String in (* just for easier reading *)

  let rec handle_helper stmts cur_exp n = match cur_exp with

      (* (A.String as st, SCall(fn, args)) -> *)
      (* let cur_tmp = "tmp" ^ (string_of_int n) in *)
      (* assign cur_tmp (st, SCall(fn, args)) :: stmts, (st,
SId(SFinalID(cur_tmp))), n + 1 *)

    (st, SCall(fn, args)) ->
      let foldable_helper (pres, es, n) exp =
        let cur_pres, cur_e, n' = handle_helper [] exp n in
        pres @ cur_pres, cur_e :: es, n'
      in
      let pres, es, n' = List.fold_left foldable_helper (stmts,[],n) args in
      (match st with
         String ->
         let cur_tmp = "tmp" ^ (string_of_int n') in
         assign cur_tmp (st, SCall(fn, List.rev es)) :: pres, (st,
SId(SFinalID(cur_tmp))), n' + 1
       | _ -> pres, (st, SCall(fn, List.rev es)), n'
      )

    (* All binary assignments should have been converted to = in semant *)
    | (A.String, SBinassop(s1, _, s2)) -> let new_stmts, s2', n' =
handle_helper stmts s2 n
      in new_stmts, (String, SBinassop(s1, Assign, (String,
SCall("cnet_strcpy", [String, SId(s1); (s2')])))), n'

    | (fst_typ, SBinop((t1, e1), op, (t2, e2))) -> (match (t1, t2) with

          (String, String) ->
          (match op with
             Add -> (* "hello" + "there" *)
             let cs1, e1', n' = handle_helper stmts (t1, e1) n
             in
             let cs2, e2', n'' = handle_helper cs1 (t1,e2) n'
             in
             let cur_tmp = "tmp" ^ (string_of_int n'') in
             assign cur_tmp (String, SCall("cnet_strcat", [e1'; e2'])) :: cs2,
```

```
                (String, SId(SFinalID(cur_tmp))), n'' + 1
           | Eq ->
             let cs1, e1', n' = handle_helper stmts (t1, e1) n
             in
             let cs2, e2', n'' = handle_helper cs1 (t1,e2) n'
             in
             let cur_tmp = "tmp" ^ (string_of_int n'') in
             let cur_exp = (Int, SCall("cnet_strcmp", [e1'; e2'])) in
             let cur_ass = SVdecl_ass({vtyp=Int; vname = cur_tmp}, cur_exp) in
             cur_ass :: cs2,
             (Int, SId(SFinalID(cur_tmp))), n'' + 1
           | _ -> semant_err ("[COMPILER BUG] only + should be allowed on two
strings (handle_strings)"))

        | (String, Int) | (Int, String) ->
          let the_str, the_int = (if t1 = String then e1, e2 else e2, e1) in
          (match op with
             Mul ->
             let cs1, the_str', n' = handle_helper stmts (String, the_str) n
in
             let cur_tmp = "tmp" ^ (string_of_int n') in
             assign cur_tmp (String, SCall("cnet_strmult", [the_str'; Int,
the_int ])) :: cs1 ,
             (String, SId(SFinalID(cur_tmp))), n' + 1
           | _ -> semant_err "[COMPILER BUG] only * should be allowed on
string-int (hanlde_strings)")

        | _ -> [], (fst_typ, SBinop((t1, e1), op, (t2, e2))), n
      )

    |(A.String, x) -> stmts , (A.String, x), n
    | _ -> stmts, cur_exp, n
  in
  let pre_stmts, new_exp, _  = handle_helper [] sexp 1000 in
  let convert_to_free = function
      SVdecl_ass({vtyp=String; vname=vn}, _) -> SDelete(String,
SId(SFinalID(vn)))
    | SVdecl_ass({vtyp=Int; vname=vn}, _) -> SExpr(Void, SNoexpr)
    | _ -> semant_err ("[COMPILER BUG] convert_to_free not setup properly")
  in
  let l = List.rev pre_stmts in
  let free_stmts = List.map convert_to_free l in
  l , new_exp , free_stmts
;;

let strip_decls dl =
  let strip_helper = function
    SVdecl_ass({vtyp=t; vname=vn}, exp) ->
    SExpr(t, SBinassop(SFinalID(vn), Assign, exp))
    | _ ->
      semant_err "[COMPILER BUG] strip_decls passed something other than a
declaration_assign list"
  in
  List.map strip_helper dl
```

```
;;



(* the built-in variables in cnet that cannot be declared by users *)
let builtin_vars =
  let add_builtinvar m vd = StringMap.add vd.vname vd m in
  List.fold_left add_builtinvar StringMap.empty
    [
      {vname="stdout"; vtyp=File};
      {vname="stdin"; vtyp=File}
    ]
;;

(* the built-in functions in cnet that cannot be declared by users *)
let builtin_funcs, builtin_funcs_l =
  let add_bind (map, l) (return_type, name, params) =
    let f = { t = return_type; name = name; parameters = params; locals = [];
body = [] }
    in
        StringMap.add name f map, f :: l
  in List.fold_left add_bind (StringMap.empty, [])
    [
      (* I/O *)
      (* Sockets *)
      (Socket, "user_nopen", [(String, "host"); (Int, "port"); (String,
"protocol"); (String, "type")]);
      (Socket, "naccept", [(Socket, "sock")]);
      (Int, "writeln", [(Socket, "f"); (String, "s")]);

      (Int, "write", [(Socket, "sock"); (String, "s")]);
      (String, "readln", [(Socket, "sock")]);
      (String, "readall", [(Socket, "sock")]);
      (String, "read", [(Socket, "sock"); (Int, "len")]);
      (Int, "error", [(Socket, "s")]);

      (* Files *)
      (File, "user_fopen", [(String, "name"); (String, "mode");]);
      (Int, "writeln", [(File, "f"); (String, "s")]);
      (Int, "nwrite", [(File, "f"); (String, "s"); (Int, "num")]);
      (String, "readln", [(File, "f")]);
      (String, "readall", [(File, "f")]);
      (Int, "error", [(File, "f")]);


      (* Strings *)
      (String, "cnet_strcpy", [(String, "t"); (String, "s")]);
      (String, "cnet_strmult", [(String, "t"); (Int, "i")]);
      (String, "cnet_strcat", [(String, "t"); (String, "s")]);
      (Int, "cnet_strcmp", [(String, "t"); (String, "s")]);
      (Int, "slength", [(String, "s")]);
      (Float, "tofloat", [(String, "s")]); (* float of string *)
      (Int, "toint", [(String, "s")]); (* int of string *)
      (String, "user_soi", [(Int, "i")]); (* string of int *)
```

```
      (String, "upper", [(String, "t")]);
      (String, "lower", [(String, "t")]);
      (String, "substring", [(String, "t"); (Int, "start"); (Int, "end")]);
      (String, "reverse", [(String, "t")]);
      (Char, "find_char", [(String, "t"); (Char, "c")]);
      (Void, "split", [(String, "t"); (String, "delim"); (Array(String),
"dest")]);
      (Char, "charat", [(String, "s"); (Int, "i")]);

      (* Arrays *)
      (Int, "alength", [((Array(Void)), "s")]);

      (* Cnet *)
      (* (Int, "cnet_free", [(String, "s")]); *)
      (Int, "cnet_free", [(Socket, "s")])
    ]
;;


(* sast version of built-in functions *)
let sbuiltin_funcs_l =
  List.map
    (fun {t=ty; name=n ; parameters=params; body=_; locals: _} ->
       {styp=ty; sfname=n; sparameters=params; sbody=[]}) builtin_funcs_l
;;

(* Codegen utils *)
(* Changes the format of an sast program, which is a list of sdecls, to one
the
 * codegen can accept, which is a tuple of lists of vdecls, struct_decls and
 * fdecls
 *)
let decompose_program (sprog : sdecl list) =
  let helper (vdecls, strct_decls, fdecls) decl = match decl with
    | SGVdecl_ass (vd, v) -> ((vd, v) :: vdecls, strct_decls, fdecls) (* TODO:
handle SGVdecl_ass properly *)
    | SSdecl(sd) -> (vdecls, sd :: strct_decls, fdecls)
    | SFdecl(fd) -> match fd.sfname with
                    "main" ->
                      let new_params = if (fd.sparameters = []) then
[(Array(String), "__(*_*)__")] (*Fake name that user cannot reference*)
                                       else fd.sparameters in
                      let user_main =
{styp=fd.styp;sfname="user_main";sparameters=new_params;sbody=fd.sbody} in
                         (vdecls, strct_decls, user_main :: fdecls)
                    | _      -> (vdecls, strct_decls,fd::fdecls)
  in
  List.fold_left helper ([], [], []) sprog


(* the built-in structs in cnet. These MUST be in exact conjunction with those
 * declared in the libcnet/*.c and libcnet/*.h source files
 *)
let builtin_structs_l =
```

```
  let vd t n = {vtyp=t;vname=n} in
    [
      (* Some of the String types here are actually just void *s that will be
       * cast later.
       *)
      {sname="string"; members=[vd String "stub"; vd String "data"; vd Int
"length"]};
      {sname="array"; members=[vd String "stub"; vd String "data"; vd Int
"length"; vd Int "i_t"]};
      {sname="cnet_file"; members=[vd String "stub"; vd String "f"; vd Int
"io_type"]};
      {sname="cnet_socket"; members=[vd String "stub"; vd String "f"; vd Int
                                     "io_type"; vd Int "fd"; vd Int "port";
vd
                                     Int "type"; vd String "addr"]}
    ]
;;

let builtin_structs =
  let add_builtin_strct m s = StringMap.add s.sname s m in
  List.fold_left add_builtin_strct StringMap.empty
    builtin_structs_l
;;


let mem_to_idx sd member =
  let rec helper n l = match l with
    hd :: tl when hd.vname = member -> n
    | hd :: tl -> helper (n + 1) tl
  in
  helper 0 sd.members
;;
```

runtests.sh

```
#!/bin/sh

# Regression testing script for C-net.
#

# Set time limit for all operations
ulimit -t 30

globallog=runtests.log
rm -f $globallog
error=0
globalerror=0

intergrationtester="./ccnet -b"
```

```
integrationtests="tests/integration/test-*.cnet tests/integration/fail-*.cnet"

parsertester="./ccnet -a"
parsertests="tests/parser/test-*.cnet tests/parser/fail-*.cnet"

scannertester="./ccnet -t"
scannertests="tests/scanner/test-*.cnet tests/scanner/fail-*.cnet"

semantictester="./ccnet -s"
semantictests="tests/semant/test-*.cnet tests/semant/fail-*.cnet"

keep=0

Usage() {
    echo "Usage: runtests.sh [options] [.cnet files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
      echo "FAILED"
      error=1
    fi
    echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
      SignalError "$1 differs"
      echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
      SignalError "$1 failed on $*"
      return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
```

```
      SignalError "failed: $* did not report an error"
      return 1
    }
    return 0
}

Check() {
    error=0
    basename=$(basename -s ".cnet" $1)
    reffile=`echo $1 | sed 's/.cnet$//'`
    basedir="$(dirname $1)/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.out" &&


    Run $2 $1 ">" "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
      if [ $keep -eq 0 ] ; then
          rm -f $generatedfiles
      fi
      echo "OK"
      echo "###### SUCCESS" 1>&2
    else
      echo "###### FAILED" 1>&2
      globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                            s/.cnet//'`
    reffile=`echo $1 | sed 's/.cnet$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
```

```
    RunFail $2 $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
        if [ $keep -eq 0 ] ; then
            rm -f $generatedfiles
        fi
        echo "OK"
        echo "###### SUCCESS" 1>&2
    else
        echo "###### FAILED" 1>&2
        globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
      k) # Keep intermediate files
          keep=1
          ;;
      h) # Help
          Usage
          ;;
    esac
done

shift `expr $OPTIND - 1`


for file in $scannertests
do
    case $file in
      *test-*)

          Check $file "$scannertester" 2>> $globallog
          ;;
      *fail-*)
          CheckFail $file "$scannertester" 2>> $globallog
          ;;
      *)
          echo "unknown file type $file"
          globalerror=1
          ;;
    esac
done

for file in $parsertests
do
    case $file in
      *test-*)
          Check $file "$parsertester" 2>> $globallog
          ;;
```

```
      *fail-*)
          CheckFail $file $parsertester 2>> $globallog
          ;;
      *)
          echo "unknown file type $file"
          globalerror=1
          ;;
    esac
done

for file in $semantictests
do
    case $file in
    *test-*)
        Check $file "$semantictester" 2>> $globallog
        ;;
    *fail-*)
        CheckFail $file "$semantictester" 2>> $globallog
        ;;
    *)
        echo "unknown file type $file"
        globalerror=1
        ;;
    esac
done

for file in $integrationtests
do
    case $file in
      *test-*)
          './ccnet' $file
          Check ${file%.cnet} 2>> $globallog
          rm ${file%.cnet}
          ;;
      *fail-*)
          # './ccnet' $file
          # CheckFail $file ${file%.cnet}.exe 2>> $globallog
          ;;
      *)
          echo "unknown file type $file"
          globalerror=1
          ;;
    esac
done

exit $globalerror
```

Test Files

tests/parser/fail-array.cnet

```
int main() {
    // Syntax error because no arguments passed for initalization
    // Correct syntax should be new int[5];
    int[] c = new int[5]{};
}
```

tests/parser/fail-array.err

```
Syntax error on line 4 near }
```

tests/semant/fail-arr-type-mismatch.cnet

```
int main() {
    int[] x = new int[10];
    x[0] = 1;
    x[1] = 1.0;
    return 0;
}
```

tests/semant/fail-arr-type-mismatch.err

```
Semantic error: illegal assignment int = float in x[1] = 1.
```

tests/semant/fail-assign2.cnet

```
int main ()
{
    int a;
    float b;
    b = 4;
}
```

tests/semant/fail-assign2.err

```
Semantic error: illegal assignment float = int in b = 4
```

tests/semant/fail-bool1.cnet

```
int main() {
    int a = 1;
    string b = "b";

    if (b) {b;}

    return 0;
}
```

tests/semant/fail-bool1.err

```
Semantic error: expected integer expression in b
```

tests/semant/fail-bool2.cnet

```
int main() {
    float a;

    while (a) {;}

    return 0;
}
```

tests/semant/fail-bool2.err

```
Semantic error: expected integer expression in a
```

tests/semant/fail-bool3.cnet

```
int main() {
    float a;

    for (;a;) {;}

    return 0;
}
```

tests/semant/fail-bool3.err

```
Semantic error: expected integer expression in a
```

tests/parser/fail-braces-unclosed.cnet

```
//testing unclosed braces in function control flows

int main()
{
    if(z>10)
    {
        //unclosed braces after if
        if(z < 20){z=50; else {z=0;}
    }
}
```

tests/parser/fail-braces-unclosed.err

```
Syntax error on line 8 near else
```

tests/semant/fail-char-op.cnet

```
int main() {
    char a = 'a';
    char b = 'b';
    a + b * a / b;
}
```

tests/semant/fail-char-op.err

```
Semantic error: illegal binary operator char * char in b * a
```

tests/parser/fail-commas-missing.cnet

```
//testing missing commas in function arguments


int gcd(int a int b){
    while(a!=b) {
        return a;
    }
}
```

tests/parser/fail-commas-missing.err

```
Syntax error on line 4 near int
```

tests/scanner/fail-comment-unclosed.cnet

```
// test unclosed multi-line comment

int main(int argc){
    /* This is the beginning of the comment
}
```

tests/scanner/fail-comment-unclosed.err

```
Scanner error: reached end of file with an unclosed multiline comment
```

tests/semant/fail-delete.cnet

```
int main(){
    string a = "hello";
    delete a;

}
```

tests/semant/fail-delete.err

```
Semantic error: illegal identifier for delete: [string a]. Identifier should be of type Struct or Array
```

tests/semant/fail-dup-func2.cnet

```
int sum(int a, int b) {
    return a + b;
}

int sum(int a, int b, int c) {
    return a + b + c;
}

int main(int argc) {
    int a = 1;
    int b = 2;

    sum(a, b, c);
}
```

tests/semant/fail-dup-func2.err

```
Semantic error: duplicate function sum
```

tests/semant/fail-dup-func.cnet

```
int main() {
    return 0;
}
int main() {
    return 0;
}
```

tests/semant/fail-dup-func.err

```
Semantic error: duplicate function main
```

tests/semant/fail-duplicate-member.cnet

```
struct my_struct {
 int x;
 int y;
};

struct my_struct2 {
 int x;
 int x;
};

int main()
{
 return 0;
}
```

tests/semant/fail-duplicate-member.err

```
Semantic error: duplicate x
```

tests/semant/fail-duplicate-struct.cnet

```
struct my_struct {
 float x;
 string y;
};

struct my_struct2 {
 float y;
};
struct my_struct {
 file y;
};


int main()
{
 return 0;
}
```

tests/semant/fail-duplicate-struct.err

```
Semantic error: Duplicate declaration of struct my_struct
```

tests/semant/fail-duplicate-var-different-type.cnet

```
struct person {
 int id;
 string name;
};

int main()
{
 int[][][] y; /* valid */

 struct person[][] y;

 y;



 struct non_existent[][] z; /* error */
}
```

tests/semant/fail-duplicate-var-different-type.err

```
Semantic error: duplicate y
```

tests/integration/fail-empty.cnet
tests/integration/fail-empty.err
Syntax error on line 1 near

tests/semant/fail-fcall1.cnet

```
int sum(int a, int b) {
    return a + b;
}

int main() {
    int a = 1;
    int b = 2;

    sum(a, b, c);
}
```

tests/semant/fail-fcall1.err

```
Semantic error: expecting 2 arguments in sum(a, b, c)
```

tests/semant/fail-fcall2.cnet

```
int sum(int a, int b) {
    return a + b;
}

int main() {
    int a = 1;
    float b = 2.4;

    sum(a, b);
}
```

tests/semant/fail-fcall2.err

```
Semantic error: illegal argument found in call to sum : found float but expected int in b
```

tests/semant/fail-float-int-asn.cnet

```
int main() {
    int a = 1;
    float c = 1.2;

    c = a;
}
```

tests/semant/fail-float-int-asn.err

```
Semantic error: illegal assignment float = int in c = a
```

tests/semant/fail-float-int-op.cnet

```
int main() {
    int a = 1;
    float c = 1.2;

    c = a + c;
}
```

tests/semant/fail-float-int-op.err

```
Semantic error: illegal binary operator int + float in a + c
```

tests/semant/fail-follow-return.cnet

```
int main() {
    return 0;
    a;
}
```

tests/semant/fail-follow-return.err

```
Semantic error: nothing may follow a return
```

tests/semant/fail-for.cnet

```
int main()
{
  int i;

  for (i = 0; j < 10 ; i = i + 1) {
      stdout.writeln("Hello");
  }

  return 0;
}
```

tests/semant/fail-for.err

```
Semantic error: undeclared identifier j
```

tests/semant/fail-freturn.cnet

```
void sum() {
    return 0;
}

int main() {
    return 0;
}
```

tests/semant/fail-freturn.err

```
Semantic error: return statement in function sum has type int but expected void in 0
```

tests/scanner/fail-illegal-comment.cnet

```
/* /^
```

tests/scanner/fail-illegal-comment.err

```
Scanner error: reached end of file with an unclosed multiline comment
```

tests/scanner/fail-illegal-tokens.cnet

```
^*
```

tests/scanner/fail-illegal-tokens.err

```
Scanner error: illegal character on line 1
```

tests/semant/fail-invalid-global-function-call.cnet

```
int myfunc()
{
 return 1;
}

int x = 3 - myfunc();

int main() {

}
```

tests/semant/fail-invalid-global-function-call.err

```
Semantic error: non-constant expression used for global variable x
```

tests/semant/fail-invalid-main.cnet

```
int main(int a, int b) {
    return a + b;
}
```

tests/semant/fail-invalid-main.err

```
Semantic error: Invalid prototype of main function.
```

tests/semant/fail-missing-func.cnet

```
int main (){
    int a = 4;
    int b = 6;

    if (a + b != 10){
        return false_sum();
    } else{
        return a+b;
    }
}
```

tests/semant/fail-missing-func.err

```
Semantic error: unrecognized function false_sum
```

tests/semant/fail-missing-main.cnet

```
int num() {
    int b= 2;
    return b;
}
```

tests/semant/fail-missing-main.err

```
Semantic error: main function not found
```

tests/semant/fail-new-array.cnet

```
struct person {
    string name;
};

int main(){
    struct person john;
    struct person jane;
    int[][] myInts = new int[4]{1,2,3,4};
    string[] myStrings = new string[2]{"hello", "world"};
    struct person[] myPeople = new struct person[2]{john, jane};
}
```

tests/semant/fail-new-array.err

```
Semantic error: illegal assignment int[][] = int[] in myInts = new int[4] = {1, 2, 3, 4}
```

tests/semant/fail-new-assign.cnet

```
struct Dimension {
    float width;
    float height;
};

struct Shape {
    struct Dimension dimension;
    string type;
};

int main() {
    struct Shape circle = new struct Shape;
    circle.type = "circle";
    circle.dimension.width = 5.;

    circle = new struct Dimension;
}
```

tests/semant/fail-new-assign.err

```
Semantic error: illegal assignment struct Shape = struct Dimension in circle = new struct Dimension
```

tests/semant/fail-new-struct.cnet

```
// Testing invalid new struct expression
struct Dimension {
    float width;
    float height;
};

struct Shape {
    struct Dimension dimension;
    string type;
};

int main() {
    // The following line should raise a semantic error Exception
    struct Shape circle = new struct shape;
    circle.type = "circle";
    circle.dimension.width = 5.;
}
```

tests/semant/fail-new-struct.err

```
Semantic error: invalid new expression: type [struct shape] doesn't exist
```

tests/semant/fail-no-main.cnet

```
int mainy(int a, int b) {
    return a + b;
}
```

tests/semant/fail-no-main.err

```
Semantic error: main function not found
```

tests/semant/fail-nonstruct-dot.cnet

```
int main()
{
 int[][] hello;

 hello.y;
}
```

tests/semant/fail-nonstruct-dot.err

```
Semantic error: dot operator not allowed on variable hello of type int[][]
```

tests/semant/fail-operator.cnet

```
int foo(int c, float d)
{
  int c;
  float d;
  return c+d;
}

int main ()
{
    int d = 5;
    float f = 4.0;
    foo(d,f);
}
```

tests/semant/fail-operator.err

```
Semantic error: illegal binary operator int + float in c + d
```

tests/parser/fail-paren-missing.cnet

```
//testing unclosed parenthesis in operator

int main()
{
	int b;
	b = (10 * (b);
	printf(b);
}
```

tests/parser/fail-paren-missing.err

```
Syntax error on line 6 near ;
```

tests/semant/fail-return2.cnet

```
int main (){
    int a = 4;
    int b = 6;

    if (a + b != 10){
        return ("incorrect sum");
    } else{
        return a+b;
    }
}
```

tests/semant/fail-return2.err

```
Semantic error: return statement in function main has type string but expected int in "incorrect sum"
```

tests/semant/fail-return.cnet

```
int main()
{
  return;
}
```

tests/semant/fail-return.err

```
Semantic error: return statement in function main has type void but expected int in
```

tests/semant/fail-scope-basic-block.cnet

```
int id(int x) {return x;}
int main()
{
  {
     int x = 5;}
  {
     id(x);
  }
  return 0;
}
```

tests/semant/fail-scope-basic-block.err

```
Semantic error: undeclared identifier x
```

tests/parser/fail-semi-missing.cnet

```
//test missing semicolon in function definition

int main()
{
    int b;
    b=10*b
    printf(b);
}
```

tests/parser/fail-semi-missing.err

```
Syntax error on line 7 near printf
```

tests/scanner/fail-string.cnet

```
// testing string
"+rediet
```

tests/scanner/fail-string.err

```
Scanner error: illegal character in string literal on line 2
```

tests/parser/fail-strings.cnet

```
int main() {
    string a = "abc
    ";
}
```

tests/parser/fail-strings.err

```
Scanner error: illegal character in string literal on line 2
```

tests/semant/fail-str-int-asn.cnet

```
int main() {
    int a;
    a = "hello";
}
```

tests/semant/fail-str-int-asn.err

```
Semantic error: illegal assignment int = string in a = "hello"
```

tests/semant/fail-str-int-op.cnet

```
int main() {
    string c = "hello";
    int a = 4;
    c + a;
}
```

tests/semant/fail-str-int-op.err

```
Semantic error: illegal binary operator string + int in c + a
```

tests/semant/fail-str-op1.cnet

```
int main () {
    string a = "a";
    a / "a";
}
```

tests/semant/fail-str-op1.err

```
Semantic error: illegal binary operator string / string in a / "a"
```

tests/semant/fail-str-op2.cnet

```
int main () {
    string a = "a";
    a - "a";
}
```

tests/semant/fail-str-op2.err

```
Semantic error: illegal binary operator string - string in a - "a"
```

tests/semant/fail-struct-invalid-struct-member.cnet

```
struct person {
 struct pid a;
};


int main()
{
 return 0;
}
```

tests/semant/fail-struct-invalid-struct-member.err

```
Semantic error: a has unrecognized struct type [struct pid]
```

tests/semant/fail-unary-minus.cnet

```
int main() {
    string a = "hi";

    int b = -a;
}
```

tests/semant/fail-unary-minus.err

```
Semantic error: illegal unary operator -string in -a
```

tests/semant/fail-unary-not.cnet

```
int main() {
    char a = 'a';
    int b = !a;
}
```

tests/semant/fail-unary-not.err

```
Semantic error: illegal unary operator !char in !a
```

tests/semant/fail-undeclared-id.cnet

```
int main() {
    int a = 1;
    a = b + a;
    return 0;
}
```

tests/semant/fail-undeclared-id.err

```
Semantic error: undeclared identifier b
```

tests/semant/fail-unknown-func.cnet

```
int main(){
    string a = "abcdef";
    slength(a);
    return 0;
}
```

tests/semant/fail-unknown-func.err

```
Semantic error: unrecognized function slength
```

tests/scanner/fail-unmatched-quote.cnet

```
""
"
```

tests/scanner/fail-unmatched-quote.err

```
Scanner error: unmatched quote on line 2
```

tests/semant/fail-unrecognized-func1.cnet

```
int main() {
    int a = 1;
    int b = double(a);
    return 0;
}
```

tests/semant/fail-unrecognized-func1.err

```
Semantic error: unrecognized function double
```

tests/semant/fail-unrecognized-func2.cnet

```
void my_print_float(float a){
    printf(a);
}

int main() {
    int a = 4;
    int b;
    b = 5;
    // int b = 5;
    a = 4;
    my_print_float(5.1);
    b = a + b;
    return  b;
  //   return a;
}
```

tests/semant/fail-unrecognized-func2.err

```
Semantic error: unrecognized function printf
```

tests/semant/fail-void-local-decl.cnet

```
int main()
{
 void x;
}
```

tests/semant/fail-void-local-decl.err

```
Semantic error: x is a void type, which is illegal
```

tests/semant/fail-while.cnet

```
int main ()
{
    int i =0;
    while (i < 2){
        i = i+1;
        foo();
    }
}
```

tests/semant/fail-while.err

```
Semantic error: unrecognized function foo
```

tests/parser/fail-wrong-struct.cnet

```
struct my_struct {
 int i;
 float f;
 string s;
}


int main()
{
 return 0;
}
```

tests/parser/fail-wrong-struct.err

```
Syntax error on line 10 near int
```

tests/scanner/test-array-declarations.cnet

```
int[] a = new int[3];
x[0] = 1;
x[1] = 2;
x[2] = 3;

int[] b = new int[]{1,2,3,4};
```

tests/scanner/test-array-declarations.out

```
INT
LBRACKET
RBRACKET
ID(a)
ASSIGN
NEW
INT
LBRACKET
INTLIT(3)
RBRACKET
SEMI
ID(x)
LBRACKET
INTLIT(0)
RBRACKET
ASSIGN
INTLIT(1)
SEMI
ID(x)
LBRACKET
INTLIT(1)
RBRACKET
ASSIGN
INTLIT(2)
SEMI
ID(x)
LBRACKET
INTLIT(2)
RBRACKET
ASSIGN
INTLIT(3)
SEMI
INT
LBRACKET
RBRACKET
ID(b)
ASSIGN
NEW
INT
LBRACKET
RBRACKET
LBRACE
INTLIT(1)
COMMA
INTLIT(2)
```

COMMA
INTLIT(3)
COMMA
INTLIT(4)
RBRACE
SEMI
EOF

tests/parser/test-arrays.cnet

```
int main() {
    int[] a = new int[3];
    x[0] = 1;
    x[1] = 2;
    x[2] = 3;

    int[] b = new int[4]{1,2,3,4};
    int[] c = new int[5]{1};
}
```

tests/parser/test-arrays.out

```
int main()
{
 int[] a = new int[3] = {};
 x[0] = 1;
 x[1] = 2;
 x[2] = 3;
 int[] b = new int[4] = {1, 2, 3, 4};
 int[] c = new int[5] = {1};
}
```

tests/semant/test-array-struct-mix-basic.cnet

```
struct mstruct {
 float id;
};

int main(){

 int x;
 while(1) {
  struct mstruct[][] x;

  x[0][0].id;
 }

 return 0;
}
```

tests/semant/test-array-struct-mix-basic.out

```
struct mstruct {
 float id;
};

int main()
{
int x = (int : 0);
{
while ((int : 1)) {
struct mstruct[][] x = (struct mstruct[][] : );
(float : x[(int : 0)][(int : 0)].id);
}
}
return (int : 0);
}
```

tests/semant/test-arr-func-param.cnet

```
int print_arr(int[] a) { return 0; }

int main() {
    int[] x = new int[10];
    print_arr(x);

    return 0;
}
```

tests/semant/test-arr-func-param.out

```
int print_arr(int[] a)
{
return (int : 0);
}

int main()
{
{
int[] x = (int[] : );
(int[] : x=(int[] : new int[(int : 10)] = {}));
}
(int : print_arr((int[] : x)));
return (int : 0);
}
```

tests/parser/test-assign.cnet

```
int main(){int b;b=10*b;printf(b);}
```

tests/parser/test-assign.out

```
int main()
{
 int b;
 b = 10 * b;
 printf(b);
}
```

tests/scanner/test-basic-tokens.cnet

```
[] {} ()
, ; ""

+ - * /

= += -= == != ! < <= > >= && || .

if else for while break continue

int float char string socket struct void TCP UDP

new delete

return
```

tests/scanner/test-basic-tokens.out

```
LBRACKET
RBRACKET
LBRACE
RBRACE
LPAREN
RPAREN
COMMA
SEMI
STRLIT()
PLUS
MINUS
TIMES
DIVIDE
ASSIGN
PLUSEQ
MINUSEQ
EQ
NEQ
NOT
LT
LEQ
GT
GEQ
AND
OR
DOT
IF
ELSE
FOR
WHILE
BREAK
CONTINUE
INT
FLOAT
CHAR
STRING
SOCKET
```

STRUCT
VOID
ID(TCP)
ID(UDP)
NEW
DELETE
RETURN
EOF

tests/parser/test-break-continue.cnet

```
int main ()
{
    int x;
    x = 0;
    while (x < 5)
    {
        if (x == 4)
        {
            break;
        }
        if (x == 0) {
            continue;
        }
        x = x + 1;
    }
    printf("%d", x);
}
```

tests/parser/test-break-continue.out

```
int main()
{
 int x;
 x = 0;
 while (x < 5)
 {
  if (x == 4)
  {
   break;  }
  if (x == 0)
  {
   continue;  }
  x = x + 1;
 }
 printf("%d", x);
}
```

tests/semant/test-builtin-funcs.cnet

```
int main(string[] args)
{
 int argc = args.alength();

 int arg1len = args[0].slength();

 file arg_count = fopen("argl", "w");
 socket send = nopen("www.google.com", 80, "tcp", "connect");

 arg_count.writeln(soi(arg1len));

 return 0;
}
```

tests/semant/test-builtin-funcs.out

```
int main(string[] args)
{
{
int argc = (int : 0);
(int : argc=(int : alength((string[] : args))));
}
{
int arg1len = (int : 0);
(int : arg1len=(int : slength((string : args[(int : 0)]))));
}
{
file arg_count = (file : );
(file : arg_count=(file : fopen((string : "argl"), (string : "w"))));
}
{
socket send = (socket : );
(socket : send=(socket : nopen((string : "www.google.com"), (int : 80), (string : "tcp"), (string : "connect"))));
}
{
string tmp1000 = (string : soi((int : arg1len)));
(int : writeln((file : arg_count), (string : tmp1000)));
delete (string : tmp1000);
}
return (int : 0);
}
```

tests/scanner/test-chars.cnet

```
'\r'
'\f'
'\t'
'\n'
'\0'
'\''
```

tests/scanner/test-chars.out

```
CHARLIT(13)
CHARLIT(12)
CHARLIT(9)
CHARLIT(10)
CHARLIT(0)
CHARLIT(47)
EOF
```

tests/scanner/test-char-tokens.cnet

```
char a = 'c';
```

tests/scanner/test-char-tokens.out

```
CHAR
ID(a)
ASSIGN
CHARLIT(99)
SEMI
EOF
```

tests/parser/test-comment.cnet

```
// this program tests comments

//  this line tests spaced comments


/* this tests multi line
comments for a c-net program */

/*
this
tests
spaced
multi
line
comments

        */

int main ()
{
    printf(a);
}
```

tests/parser/test-comment.out

```
int main()
{
    printf(a);
}
```

tests/scanner/test-comments.cnet

```
// this is single line comment
// king //king
/* this is a multiline comment
because it fills more than one
line */
```

tests/scanner/test-comments.out

```
EOF
```

tests/scanner/test-control-flow.cnet

```
if (5 < 15 || 4 > 5) {
  return true;
} else {
     return false;
}

int x = 5;
while (x < 15) {
  x += 5;
}

for (int i =16; x < i; ) {x -= 1;}
```

tests/scanner/test-control-flow.out

```
IF
LPAREN
INTLIT(5)
LT
INTLIT(15)
OR
INTLIT(4)
GT
INTLIT(5)
RPAREN
LBRACE
RETURN
ID(true)
SEMI
RBRACE
ELSE
LBRACE
RETURN
ID(false)
SEMI
RBRACE
INT
ID(x)
ASSIGN
INTLIT(5)
SEMI
WHILE
LPAREN
ID(x)
LT
INTLIT(15)
RPAREN
LBRACE
ID(x)
PLUSEQ
INTLIT(5)
SEMI
RBRACE
FOR
```

LPAREN
INT
ID(i)
ASSIGN
INTLIT(16)
SEMI
ID(x)
LT
ID(i)
SEMI
RPAREN
LBRACE
ID(x)
MINUSEQ
INTLIT(1)
SEMI
RBRACE
EOF

tests/semant/test-delete.cnet

```
struct person {
    string name;
};

int main(){
    struct person john;
    struct person jane;
    int[] myInts;
    myInts = new int[4]{1,2,3,4};
    string[] myStrings = new string[2]{"hello", "world"};
    struct person[] myPeople = new struct person[2]{john, jane};
    delete myPeople[0];
    delete myPeople;
    delete john;
    delete myInts;

    return 0;
}
```

tests/semant/test-delete.out

```
struct person {
 string name;
};

int main()
{
struct person john = (struct person : );
struct person jane = (struct person : );
int[] myInts = (int[] : );
(int[] : myInts=(int[] : new int[(int : 4)] = {(int : 1), (int : 2), (int : 3), (int : 4)}));
{
string[] myStrings = (string[] : );
(string[] : myStrings=(string[] : new string[(int : 2)] = {(string : "hello"), (string : "world")}));
}
{
struct person[] myPeople = (struct person[] : );
(struct person[] : myPeople=(struct person[] : new struct person[(int : 2)] = {(struct person : john), (struct pe
}
delete (struct person : myPeople[(int : 0)]);
delete (struct person[] : myPeople);
delete (struct person : john);
delete (int[] : myInts);
return (int : 0);
}
```

tests/semant/test-fcall1.cnet

```
int main() {
    int a = 5;
    int b = y(x(a));

    return 0;
}

int x(int b) {return b;}
int y(int c) {return c;}
```

tests/semant/test-fcall1.out

```
int main()
{
{
int a = (int : 0);
(int : a=(int : 5));
}
{
int b = (int : 0);
(int : b=(int : y((int : x((int : a))))));
}
return (int : 0);
}

int x(int b)
{
return (int : b);
}

int y(int c)
{
return (int : c);
}
```

tests/integration/test-file-io1.cnet

```
int main(){

  file f = fopen("_tags", "rb");
  int i = 0;
  while (i < 9){
    string line = f.readln();
    // write extra line because of readln and writeln double usage
    stdout.writeln(line);
    i += 1;
  }
  return 0;
}
```

tests/integration/test-file-io1.out

```
# Include the llvm and llvm.analysis packages while compiling

true: package(llvm), package(llvm.analysis)



# Enable almost all compiler warnings

true : warn(+a-4-42)



# Enble use of Str package

true: use_str
```

tests/scanner/test-float-complex.cnet

```
1.
0.5e-15
.3e+3
.2
1e5
3.5e-4
```

tests/scanner/test-float-complex.out

```
FLOATLIT(1.000000)
FLOATLIT(0.000000)
FLOATLIT(300.000000)
FLOATLIT(0.200000)
FLOATLIT(100000.000000)
FLOATLIT(0.000350)
EOF
```

tests/integration/test-func1.cnet

```
int main() {
    int a;
    a = 4;
    stdout.writeln(soi(5*a));
    return 0;
}
```

tests/integration/test-func1.out

```
20
```

tests/semant/test-func.cnet

```
int main() {
    int a;
    a = 4;
    return 5*4;
}
```

tests/semant/test-func.out

```
int main()
{
int a = (int : 0);
(int : a=(int : 4));
return (int : (int : 5) * (int : 4));
}
```

tests/semant/test-global-basic-int.cnet

```
int x = 9 * 5;
int y = 3 - 9;
int z = 4 * 9 - 1 - 12 % 5;


int a = 1 && 0;
int b = 1 || 0;


int c = 15 == 9;
int d = 2 == 2;

int main () {

 return 0;
}
```

tests/semant/test-global-basic-int.out

```
int x = (int : 45);
int y = (int : -6);
int z = (int : 33);
int a = (int : 0);
int b = (int : 1);
int c = (int : 0);
int d = (int : 1);
int main()
{
return (int : 0);
}
```

tests/integration/test-hello-world.cnet

```
int main()
{
 stdout.writeln("Hello World!");

 return 0;
}
```

tests/integration/test-hello-world.out

```
Hello World!
```

tests/semant/test-if1.cnet

```
int main(string[] argv) {
    int a = 9;
    int b = 4;
    if (a > b) {
        ;/* stdout.println("a is greater than b"); */
    }
    else
        ;/* stdout.println("a is less or equal to b"); */
    int c = 9;

    return c;

}
```

tests/semant/test-if1.out

```
int main(string[] argv)
{
{
int a = (int : 0);
(int : a=(int : 9));
}
{
int b = (int : 0);
(int : b=(int : 4));
}
{
if ((int : (int : a) > (int : b)))
{
(void : );
}
else if ((int : 1))
(void : );
}
{
int c = (int : 0);
(int : c=(int : 9));
}
return (int : c);
}
```

tests/parser/test-if-ifelse-while-for.cnet

```
int main()
{
    int c;
    c=10;
    int b;
    if(c > 9) {printf("%d", c);}
    if(c>10)
    {
        a=200;
    } else {
        b=10;

        c=30;
    }
    while(b * 2.5 < c)
        {
            b=b+2;
            printf(b + 5);
        }
    for (i = 1; i < 11; i = i + 1)
    {
        printf("%d ", i);
    }
}
```

tests/parser/test-if-ifelse-while-for.out

```
int main()
{
        int c;
        c = 10;
        int b;
        if (c > 9)
        {
                printf("%d", c);
        }
        if (c > 10)
        {
                a = 200;
        }
        else
        {
                b = 10;
                c = 30;
        }
        while (b * 2.5 < c)
        {
                b = b + 2;
                printf(b + 5);
        }
        for (i = 1 ; i < 11 ; i = i + 1)        {
                printf("%d ", i);
        }
}
```

tests/integration/test-if.cnet

```
int main ()
{
    int i = 4;
    if (i>0){
        stdout.writeln("hello");
    }
    return 0;
}
```

tests/integration/test-if.out

```
hello
```

tests/parser/test-infinite-for.cnet

```
int main() {
    for (; ;)
    {
        printf("%d ", 0);
    }
}
```

tests/parser/test-infinite-for.out

```
int main()
{
 for ( ;  ; )  {
  printf("%d ", 0);
 }
}
```

tests/scanner/test-literals.cnet

```
10023
100.24
"Hello, World"
"Hello\nWorld"
"Hello'\r\t World"
'a'
'\012'
```

tests/scanner/test-literals.out

```
INTLIT(10023)
FLOATLIT(100.240000)
STRLIT(Hello, World)
STRLIT(Hello
World)
STRLIT(Hello'
  World)
CHARLIT(97)
CHARLIT(10)
EOF
```

tests/scanner/test-loops.cnet

```
//testing loops

for(count = 1; count <= num; count= COUNT+1)
{
   sum += count;
}
```

tests/scanner/test-loops.out

```
FOR
LPAREN
ID(count)
ASSIGN
INTLIT(1)
SEMI
ID(count)
LEQ
ID(num)
SEMI
ID(count)
ASSIGN
ID(COUNT)
PLUS
INTLIT(1)
RPAREN
LBRACE
ID(sum)
PLUSEQ
ID(count)
SEMI
RBRACE
EOF
```

```
struct id {
 int id;
 struct id next;
 string id_s;
};

struct person {
 struct id id;
 string name;
};

struct dozen {
 struct person[] people;
};

int main()
{

 struct dozen my_doz;

 my_doz.people = new struct person[12];

 my_doz.people[2].name = "Bob";


 struct person p;

 my_doz.people[5].id = p.id;


 my_doz.people[5] = p;


 struct person[] tmp = my_doz.people;


 tmp[1] = p;

 int[][] x = new int[][3]{1,2,3};

 x[0] = new int[12];

 x[0][2] = 12;

}
```



```
struct id {
 int id;
 struct id next;
 string id_s;
};
```

```
struct person {
 struct id id;
 string name;
};

struct dozen {
 struct person[] people;
};

int main()
{
 struct dozen my_doz;
 my_doz.people = new struct person[12] = {};
 my_doz.people[2].name = "Bob";
 struct person p;
 my_doz.people[5].id = p.id;
 my_doz.people[5] = p;
 struct person[] tmp = my_doz.people;
 tmp[1] = p;
 int[][] x = new int[][3] = {1, 2, 3};
 x[0] = new int[12] = {};
 x[0][2] = 12;
}
```

tests/parser/test-nested-for.cnet

```
int main()
{
    int a;
    int b;
    int res;
    a = 1;
    b = 1;
    res = 0;

    for(a; a < 15; a = a+1)
    {
        for(b; b < 15; b = b + 1)
        {
            res = res + b;
        }
    }
}
```

tests/parser/test-nested-for.out

```
int main()
{
 int a;
 int b;
 int res;
 a = 1;
 b = 1;
 res = 0;
 for (a ; a < 15 ; a = a + 1)  {
  for (b ; b < 15 ; b = b + 1)   {
   res = res + b;
  }
 }
}
```

tests/parser/test-nested-ifelse.cnet

```
int main()
{
    int x;
    x=92322;
    if(x>32342)
    {
        x=2000;
        if(x < 2000)
        {
            x=2000;
        }
        else
        {
            x=32342;
        }
    }
    else
    {
        x=100;
    }
}
```

tests/parser/test-nested-ifelse.out

```
int main()
{
 int x;
 x = 92322;
 if (x > 32342)
 {
  x = 2000;
  if (x < 2000)
  {
   x = 2000;
  }
  else
  {
   x = 32342;
  }
 }
 else
 {
  x = 100;
 }
}
```

tests/parser/test-nested-if.cnet

```
int main()
{
    int z;z=15;
    if(z>10)
    {
        z=30;
        if(z < 20){z=50;} else {z=0;}
    }
}
```

tests/parser/test-nested-if.out

```
int main()
{
	int z;
	z = 15;
	if (z > 10)
	{
		z = 30;
		if (z < 20)
		{
			z = 50;
		}
		else
		{
			z = 0;
		}
	}
}
```

tests/parser/test-nested-mix.cnet

```
int main()
{
    int x;
    int y;
    int c;
    int b;
    b = 0;
    c = 100;
    while(x == y)
    {
        while(b == 0)
        {
            for (b; b < 101; b = b + 1)
            {
                if (b == c)
                {
                    if (x == y)
                    {
                        printf("true");
                    }
                    printf(10);
                }
                else
                {
                    printf(5);
                    if (x != y)
                    {
                        printf("");
                    }
                    else
                    {
                        printf("false");
                    }
                }
            }
            for (; b < 101;) {printf("no");}
        }

    }
}
```

tests/parser/test-nested-mix.out

```
int main()
{
 int x;
 int y;
 int c;
 int b;
 b = 0;
 c = 100;
 while (x == y)
 {
  while (b == 0)
```

```
  {
   for (b ; b < 101 ; b = b + 1)    {
    if (b == c)
    {
     if (x == y)
     {
      printf("true");
     }
     printf(10);
    }
    else
    {
     printf(5);
     if (x != y)
     {
      printf("");
     }
     else
     {
      printf("false");
     }
    }
   }
   for ( ; b < 101 ; )    {
    printf("no");
   }
  }
 }
}
```

tests/parser/test-nested-while.cnet

```
int main()
{
    int a;
    int b;
    int res;
    a = 1;
    b = 1;
    res = 0;

    while(a<15)
    {
        while(b<15)
        {
            res = res + b;
            b = b+1;
        }
        a = a+1;
    }
}
```

tests/parser/test-nested-while.out

```
int main()
{
 int a;
 int b;
 int res;
 a = 1;
 b = 1;
 res = 0;
 while (a < 15)
 {
  while (b < 15)
  {
   res = res + b;
   b = b + 1;
  }
  a = a + 1;
 }
}
```

tests/semant/test-new-array.cnet

```
struct person {
    string name;
};

int main(){
    struct person john;
    struct person jane;
    int[] myInts;
    myInts = new int[4]{1,2,3,4};
    string[] myStrings = new string[2]{"hello", "world"};
    struct person[] myPeople = new struct person[2]{john, jane};


    return 0;
}
```

tests/semant/test-new-array.out

```
struct person {
 string name;
};

int main()
{
struct person john = (struct person : );
struct person jane = (struct person : );
int[] myInts = (int[] : );
(int[] : myInts=(int[] : new int[(int : 4)] = {(int : 1), (int : 2), (int : 3), (int : 4)}));
{
string[] myStrings = (string[] : );
(string[] : myStrings=(string[] : new string[(int : 2)] = {(string : "hello"), (string : "world")}));
}
{
struct person[] myPeople = (struct person[] : );
(struct person[] : myPeople=(struct person[] : new struct person[(int : 2)] = {(struct person : john), (struct pe
}
return (int : 0);
}
```

tests/semant/test-new-struct.cnet

```
struct Dimension {
    float width;
    float height;
};

struct Shape {
    struct Dimension dimension;
    string type;
};

int main() {
    struct Shape circle = new struct Shape;
    circle.type = "circle";
    circle.dimension.width = 5.;

    return 0;
}
```

tests/semant/test-new-struct.out

```
struct Dimension {
 float width;
 float height;
};

struct Shape {
 struct Dimension dimension;
 string type;
};

int main()
{
{
struct Shape circle = (struct Shape : );
(struct Shape : circle=(struct Shape : new Shape));
}
(string : circle.type=(string : cnet_strcpy((string : circle.type), (string : "circle"))));
(float : circle.dimension.width=(float : 5.));
return (int : 0);
}
```

tests/semant/test-scope-basic-block2.cnet

```
int main()
{
   {
      int x = 5;
      int y = 10;
      {
         {
            int y = 20;

            x+=1;

            y+=1;
         }
         int z = x + y; // outer block x and y
      }
   }
   return 0;
}
```

tests/semant/test-scope-basic-block2.out

```
int main()
{
{
{
int x = (int : 0);
(int : x=(int : 5));
}
{
int y = (int : 0);
(int : y=(int : 10));
}
{
{
{
int y = (int : 0);
(int : y=(int : 20));
}
(int : x=(int : (int : x) + (int : 1)));
(int : y=(int : (int : y) + (int : 1)));
}
{
int z = (int : 0);
(int : z=(int : (int : x) + (int : y)));
}
}
}
return (int : 0);
}
```

tests/semant/test-scope-basic-block.cnet

```
int sum(int x, int y, int z)
{
 y; /* this is the formal */
 {
  float y;
  char [][]z;

  y; /* this is the float y */

  for(x = 3 ; x > 2; x -= 5)
  {
   int []y;

   y; /* the int[] y */
   x; /* the x from the formals */
   z; /* the z from the block above */
  }
 }

 return 0;
}

int main()
{
 return 0;
}
```

tests/semant/test-scope-basic-block.out

```
int sum(int x,int y,int z)
{
(int : y);
{
float y = (float : 0.);
char[][] z = (char[][] : );
(float : y);
{
(int : x=(int : 3));
{
while ((int : (int : x) > (int : 2))) {
{
int[] y = (int[] : );
(int[] : y);
(int : x);
(char[][] : z);
}
(int : x=(int : (int : x) - (int : 5)));
}
}
}
}
return (int : 0);
}
```

```
int main()
{
return (int : 0);
}
```

tests/semant/test-scope-func-param.cnet

```
int main()
{
    int z = mul(2, x);
    return 0;
}
int add(int a, int b) {return a + b;}
int x = 1;
int mul(int x, int y){return x * y;}
```

tests/semant/test-scope-func-param.out

```
int main()
{
{
int z = (int : 0);
(int : z=(int : mul((int : 2), (int : x))));
}
return (int : 0);
}

int add(int a,int b)
{
return (int : (int : a) + (int : b));
}

int x = (int : 1);
int mul(int x,int y)
{
return (int : (int : x) * (int : y));
}
```

tests/semant/test-str-concat.cnet

```
int main() {
    string c = "a" + "A";

    return 0;
}
```

tests/semant/test-str-concat.out

```
int main()
{
{
string tmp1000 = (string : cnet_strcat((string : "a"), (string : "A")));
string c = (string : "");
(string : c=(string : cnet_strcpy((string : c), (string : tmp1000))));
delete (string : tmp1000);
}
delete (string : c);
return (int : 0);
}
```

tests/integration/test-strconcat.cnet

```
int main() {
    stdout.writeln("RED" + "alert");
    return 0;
}
```

tests/integration/test-strconcat.out

```
REDalert
```

tests/scanner/test-string.cnet

```
string a =
"abc" + "def"
```

tests/scanner/test-string.out

```
STRING
ID(a)
ASSIGN
STRLIT(abc)
PLUS
STRLIT(def)
EOF
```

tests/parser/test-strings.cnet

```
int main() {
    string a = "abc\n" + "";
    string b = "^$&Q*!)LL ";
}
```

tests/parser/test-strings.out

```
int main()
{
 string a = "abc
" + "";
 string b = "^$&Q*!)LL ";
}
```

tests/parser/test-struct-1.cnet

```
struct dimension {
    float width;
    float height;
};

struct Shape {
    struct dimension area;
    string type;
};

int main(int argc) {
    struct Shape circle;
    circle.type = "circle";
    circle.dimension.width = 5.;
}
```

tests/parser/test-struct-1.out

```
struct dimension {
 float width;
 float height;
};

struct Shape {
 struct dimension area;
 string type;
};

int main(int argc)
{
 struct Shape circle;
 circle.type = "circle";
 circle.dimension.width = 5.;
}
```

tests/scanner/test-struct-member.cnet

```
struct animal {
  string name;
  string color;
  int weight;
};
struct animal my_animal = new struct animal;

animal.name  = "goat";
animal.color = "brown";
animal.weight = 200.40;
```

tests/scanner/test-struct-member.out

```
STRUCT
ID(animal)
LBRACE
STRING
ID(name)
SEMI
STRING
ID(color)
SEMI
INT
ID(weight)
SEMI
RBRACE
SEMI
STRUCT
ID(animal)
ID(my_animal)
ASSIGN
NEW
STRUCT
ID(animal)
SEMI
ID(animal)
DOT
ID(name)
ASSIGN
STRLIT(goat)
SEMI
ID(animal)
DOT
ID(color)
ASSIGN
STRLIT(brown)
SEMI
ID(animal)
DOT
ID(weight)
ASSIGN
FLOATLIT(200.400000)
SEMI
EOF
```

tests/semant/test-struct-member-types-basic.cnet

```
struct pid {
 int id;
};

struct person {
 struct pid x;
 struct person partner;
};

int main() {
 struct person s;


 s; /* type struct person */
 s.x; /* type struct pid */
 s.x.id; /* type int */
 s.partner; /* type struct person */
 s.partner.partner; /* type struct person */
 s.partner.partner.x.id; /* type int */


 /* s.partner.partner.x.partner; ERROR */

 return 0;
}
```

tests/semant/test-struct-member-types-basic.out

```
struct pid {
 int id;
};

struct person {
 struct pid x;
 struct person partner;
};

int main()
{
struct person s = (struct person : );
(struct person : s);
(struct pid : s.x);
(int : s.x.id);
(struct person : s.partner);
(struct person : s.partner.partner);
(int : s.partner.partner.x.id);
return (int : 0);
}
```

tests/semant/test-struct-nested-member.cnet

```
struct pid {
 int pid;
 string wing;
};

struct person {
 struct pid id;
 string name;
 struct person partner;
};



int main()
{
 return 0;
}
```

tests/semant/test-struct-nested-member.out

```
struct pid {
 int pid;
 string wing;
};

struct person {
 struct pid id;
 string name;
 struct person partner;
};

int main()
{
return (int : 0);
}
```

tests/parser/test-struct.cnet

```
struct person {
    string name;
    int age;
};

int main(int argc) {
    struct person jane;
    jane.age = 15;
    jane.name = 2;
}
```

tests/parser/test-struct.out

```
struct person {
 string name;
 int age;
};

int main(int argc)
{
 struct person jane;
 jane.age = 15;
 jane.name = 2;
}
```

tests/scanner/test-var.cnet

```
int a;

// test

int y = 7;
```

tests/scanner/test-var.out

```
INT
ID(a)
SEMI
INT
ID(y)
ASSIGN
INTLIT(7)
SEMI
EOF
```

tests/parser/test-vdecl.cnet

```
int main(){int


b ;         }
```

tests/parser/test-vdecl.out

```
int main()
{
 int b;
}
```

tests/integration/test-while.cnet

```
string foo (int x){
    return soi(x);
}

int main (){
    int i = 4;
    while (i > 0){
        stdout.writeln(foo(i));
        i= i - 1;
    }
    return 0;
}
```

tests/integration/test-while.out

```
4
3
2
1
```