THEATR

An actor based language

Final Report

May 10th, 2017

Group members:
Beatrix Carroll (bac2108)
Suraj Keshri (skk2142)
Michael Lin (mbl2109)
Linda Ortega Cordoves (lo2258)

filedwld.c
filedwld.h
theatr.ml
run.sh
testall.sh

Introduction

Theatr is an Actor-based language with the goal of granting the user core actor model principles in a highly readable and straightforward syntax.

We were interested in the Actor model because it offers promising archetype for how to create a concurrent-focused programming language. Furthermore, actor paradigm as seen in Theatr encourages the development of programs that are well-written from a concurrency and fault-tolerance perspective.

With threading, locking, and separation of resources built into the language, we hope to provide users with the ability to write concurrent programs using the actor model in an intuitive manner.

Language Tutorial

Theatr uses a C-like imperative syntax along with whitespace-determined scoping as in Python. Indentation whitespace determines the scope (with exactly **4** spaces of indentation per level - not 3, or 5, but 4 - and certainly no tab characters), and the one main requirement of the program is the declaration of a **`main()`** function that acts as the entry point of execution.

Simple “Hello world!” example

Below is a short example of a **`main()`** function that prints a string to output, without using any actor functionality:

```
func main() -> int:
    print("Hello world! I'd like to be a dolphin.")
    return 0
```

Simple example using actors

What if we wanted a dolphin actor to be the one talking? The snippet below shows how to form actors and send messages to them, and having the actor act upon a message:

```
dolphin(int weight):
    print("I'm a new dolphin!")

    receive:
        say_hello():
            print("Hello world! I'm a dolphin.")
        eat(int food):
            weight = weight + food
    drop:
        print("I couldn't understand you.")

func main() -> int:
    actor emerson = new dolphin(12)
    dolphin.say_hello() | emerson
    dolphin.eat(5) | emerson
    dolphin.die() | emerson
    return 0
```

In the above example, we've defined what a **`dolphin`** actor can do in the **`dolphin(int weight)`** declaration: it prints out a statement on startup and its **`receive`** block dictates what messages it will act upon; all unrecognized messages received will trigger the dolphin to perform statements in the **`drop`** block. Theatr actors have a few key properties:

- **Actors run in independent threads asynchronously with the main() function and any other actors.**
- **Actor state variables are stored on the local thread stack - and can only be accessed and modified by that actor**
- **Messages sent to an actor are placed in that actor's message queue, and the actor sequentially processes messages one at a time, in the order received.**

The result is that concurrency is built in and much safer in Theatr, as each actor processes messages one at a time and maintains independent states. So in our example, **`emerson`** will continually check and read messages in its message queue and execute them sequentially as they arrive from **`main()`**. A few rules to note:

- Messages can be sent to the dolphin **`emerson`** by calling the message, with a `|`, and the recipient id. **`emerson`** will evaluate the messages sequentially in the order they are received.

- Messages must include the full namespace of the type of actor, by including **`<actor type>.<message name>(<optional message arguments>)`**: so a **`dolphin.eat()`** message is distinct from a **`fish.eat()`** message
- All **`dolphin`**s keep track of their own **`int weight`** variable as a state variable, and only they can access it directly
- When **`emerson`** processes a **`dolphin.die()`** message, it ends its thread of execution. Theatr provides a built-in **`<actor type>.die()`** message for every actor as stop a running actor.

The above program will produce the following output:

```
I'm a new dolphin!
Hello world! I'm a dolphin.
```

Multiple actors and passing around actor references

To showcase Theatr's main features around actor communication, let's look at an example with multiple actors. Suppose our dolphin wishes to interact with other actors, namely a fish, by eating it. We declare a new type of actor, fish:

```
fish():
    actor self
    receive:
        be_eaten():
            print("Fish: I've been eaten!")
            fish.die() | self
        instantiate(actor id):
            self = id
    drop:
        print("I couldn't understand that.")
```

Notice that the fish actor contains a state variable **`self`**, of type **`actor`**, which can be set by processing an **`instantiate()`** message. This allows someone to pass in an actor id using **`instantiate(actor id)`**, which fish will store in its local state, and which is used later in **`be_eaten()`**. This highlights another main feature of Theatr:

- **An actor id is a reference that can be passed around as a handle to send messages to that actor's message queue**.
- **Actors can send and receive handles to other actors via messages - once they have access to a handle, they can themselves then send messages to that actor**

If a scope of the program has access to an actor's handle, it is free to send messages to it. This gives rise to powerful abilities such as passing actor ids in messages in order to get actors to communicate with each other. In this case, when the fish receives a message **`be_eaten()`**, it sends a **`die()`** message to itself, terminating its own execution!

Our new dolphin declaration now consists of a **`hunt(actor f)`** message type, providing a way for dolphin to receive a handle to a fish to hunt. Upon processing a **`hunt(f)`** message, the dolphin will send a message to fish **`f`**, informing **`f`** that it has been eaten, and causing the fish to terminate itself:

```
dolphin(int weight):
    receive:
        hunt(actor f):
            print("Dolphin: thanks for the fish!")
            fish.be_eaten() | f
    drop:
        print("I couldn't understand that.")
```

Finally, our **`main()`** function kicks off the action:

```
func main() -> int:
    actor f = new fish()
    fish.instantiate(f) | f
    actor emerson = new dolphin(12)
    dolphin.hunt(f, 4) | emerson
    dolphin.die() | emerson
    return 0
```

And produces the output:

```
Dolphin: thanks for the fish!
Fish: I've been eaten! Terminating myself...
```

With actors running in independent threads asynchronously and the ability to communicate with each other using handles, the user can use Theatr to write concurrent programs using the actor model and message passing.

Language Reference Manual

Preface

Goal

Create an actor-based language with fault-tolerance that simulates the logic of Erlang but has the syntax of a language that's easily accessible to new programmers, like Python or Ruby.

Motivation

The actor model provides a good framework for reasoning about concurrency and distributed systems. The model elegantly handles concurrency issues which can be complex when implemented in a thread-based model, such as locking and mutual access to resources. Instead of multiple threads accessing a shared resource, the actor model relies on autonomous actors performing asynchronous and independent computations.

Distributed systems requires comprehensive and secure communication capabilities with its various remote processors. To achieve this, distributed systems require that processors communicate via messages and that these messages be asynchronous. In the actor model, actors communicate with each other exclusively through messages, which are stored in mailboxes and processed asynchronously.

Description

THEATR is a programming language that implements the Actor Model with fault-tolerance.

As the Actor Model specifies, actors are the basic unit of computation in our language. Upon receiving a message, any actor will be able to only perform the following three actions:

1. Create another actor
2. Send message to another actor
3. Update the internal state (specify the state in which it will be when it next receives a message)

Each actor possesses an internal state. THEATR actors will only be able to communicate with each other actors via passing messages. Each message an actor receives will be stored in its mailbox and processed asynchronously.

Programs

Our language is designed to write programs that take advantage of parallelism. Some examples of these types of programs are: chat servers, phone switches, web servers, message queues, and web crawlers. Our language is also designed for programs requiring multiple I/O computations, programs requiring strict fault tolerance, and programs requiring strict avoidance of deadlocking.

Lexical Elements

Identifiers

Identifiers can include letters, decimal digits, and the underscore character. The first character of an identifier cannot be a digit. Identifiers are case sensitive.

Keywords

These keywords are reserved for use in our language:
if, else, for, while, actor, int, double, string, char, bool, receive, drop, after, new, return, func, main, true, false.

Constants

A constant is a literal numeric or character value, such as 5 or 'm'. All constants are of a particular data type.

An **integer** constant is a sequence of digits, with an optional prefix to denote a number base. An integer constant is a sequence of digits.

A **double** constant is a value that represents a fractional (floating point) number. It consists of a sequence of digits which represents the integer (or “whole”) part of the number, a decimal point, and a sequence of digits which represents the fractional part. Either the integer part or the fractional part may be omitted, but not both.

A **character** constant is usually a single character enclosed within single quotation marks, such as 'Q'. A character constant is of type int by default.

A **string** constant is a sequence of zero or more characters, digits, and escape sequences enclosed within double quotation marks.

Separators

A separator separates tokens. White space (see next section) and tabs are separators, but they are not a token. The other separators are all single-character tokens themselves:

Separator	Usage
( )	Orders operations within an expression, groups expressions
{ }	Denotes scope of code blocks, or a definition of a struct
, (comma)	Separator for function arguments list
; (semicolon)	Separator for statements

White Space

Like Python, our language is whitespace sensitive: it defines the hierarchy of the code. The space character is also important in string and character constants and when separating tokens. Indentation determines scope: each level of indentation must be exactly **4** spaces (no tab characters allowed)

Comments

The /* */ symbols comments out entire sections of code, while // comments out a single line.

```
/*
comment
block
 */

// commented line
```

Data Types

Data type keyword	Meaning
int	Integers, 4 bytes in size
double	Real numbers in floating point notation, 8 bytes in size
bool	Return 1 or 0 in value. When invoking a boolean value, use “true” or “false”. I.e: (true && true) // returns 1
char	Represent the ASCII character set, 1 byte in size

string	Collections of characters
actor	Represents a pointer to an actor's message queue, and associated metadata

Actors:

As an actor-based language, actors are a central data type! They have the following rules enforced by the compiler:

- All actors must implement a receive block (more details below)
- All actors must have a drop block containing statements to execute when an unrecognized message is processed
- They must not have a return statement in any statement contained in the body, or the body of any receive or drop statements.

The syntax of defining an actor is:

```
[actor name] ([actor state variables, as parameters]):
	receive:
		[message name]([message paramenters]):
			[behavior upon receiving message]
	[drop:]
		[behavior upon receiving a message not accounted for in the receive block]
```

For example, to define an actor dolphin that keeps track of a name and weight, with the ability to receive eat(int number) messages:

```
dolphin(string name, int weight):
	receive:
		eat(int numFoodPellets):
			weight = weight + numFoodPellets
		runAway():
			print("So long, and thanks for all the fish!")
	drop:
		print("I didn't get that")
```

Sending messages

Provided an actor identifier, any actor or any function can send a message to that actor. To send a message, the full namespace of that message including the type of actor it is meant for, must be included, using the following syntax:

```
[actor type].[message type] | [recipient]
```

For example:

```
dolphin.eat(5) | d
```

Note that actor identifiers can be passed in as arguments, which allows actors to receive other actors to send messages to.

Expressions, Assignment, and Operators

Expressions and Assignment

An **expression** is any legal combination of symbols that represents a value. Assignments are expressions that set variables equal to the value of another expression. The syntax for assignments is:

```
[type of variable] [variable name] = [expression]
```

For example:

```
int x = 5
int dist = sqrt(x*x, y*y)
```

Assignment, Access, Function, and Actor-specific Operators

Operator	**Meaning**	**Associativity**
=	Assignment	Left
. (period)	Specifies namespace for the message to be sent. Usage: [actor type].[message name] | target	N/A
->	Denotes the return value for a named function	N/A
| (pipe)	Send message. Usage: [message type] | [target actor]	N/A

Arithmetic Operators

Operator	**Meaning**	**Associativity**
+	Addition	Left
-	Subtraction	Left

*	Multiplication	Left
/	Division	Left
%	Modulus	Left
- (unary minus)	Value of the expression multiplied by -1	Right

Comparison Operators

Operator	**Meaning**	**Associativity**
==	Equal to	N/A
!=	Not equal to	N/A
>	Greater than	N/A
<	Less than	N/A
<=	Less than or equal to	N/A
>=	Greater than or equal to	N/A

Logical Operators

Operator	**Meaning**	**Associativity**
!	Not, negation	Right
||	Or	Left
&&	And	Left

Operator Precedence

The rules of precedence used when trying to solve an expression containing multiple operators. Below is a list of types of expressions, presented in order of highest precedence first.

1. Sending messages to actors and waiting for a response.

1.1. Messages' arguments are evaluated before being sent, if the message's arguments contain a compound expression
    1.1.1. For example:
        1.1.1.1. bank_account.make_deposit(x+5) | ba
2. Function calls, and membership access operator expressions.
3. Unary operators, including logical negation, increment, decrement, unary positive, unary negative, indirection operator, address operator, and sizeof expressions. When several unary operators are consecutive, the later ones are nested within the earlier ones: !-x means !(-x).
4. Multiplication and division.
5. Addition and subtraction expressions.
    5.1.1. For example, in the expression a+b * f() the order of precedence is to call the the function f with no arguments, multiply the result by b, then add that result to a.
6. Greater-than, less-than, greater-than-or-equal-to, and less-than-or-equal-to expressions.
7. Equal-to and not-equal-to expressions.
8. Logical AND expressions.
9. Logical OR expressions.
10. All assignment expressions
11. Comma operator expressions.

Statements

The *if* Statement

You can use the `if` statement to conditionally execute part of your program, based on the truth value of a given expression. Indentation determines the attachments of any `else` or `else if` statements to the parent `if`. The syntax is as follows (note the indentation):

```
if (test)
    then-statement
else
    Else-statement
```

The *while* Statement

The while statement is a loop statement with an exit test at the beginning of the loop. The indentation of the statement denotes the scope of the while loop:

```
while (test):
    Statement
```

The *for* Statement

The for statement is a loop statement whose structure allows easy variable initialization, expression testing, and variable modification.

```
for (initialize; test; step):
    Statement
```

The variable used to track the number of loops cannot be initialized in the for-loop's initialize statement.

The *return* Statement

The return statement comes at the end of a function to denote the value that the function evaluates to.

Functions

Function Definitions

Named functions are declared using the func keyword, followed by the name identifier of the function, its argument list, and the colon character, as per the following:

```
func function_name (data_type x, data_type y) -> return_data_type:
      /* function_body */
      return value
```

For example:

```
func sqrt(int x) -> int:
      return x * x
```

Calling Functions

The syntax for calling named functions is as follows:

```
int y = function_name (int x, int y)
```

Function Parameters

Within the function body, the parameter is a local copy of the value passed into the function; you cannot change the value passed in by changing the local copy. If you wish to use the function to change the original value, then you would have to incorporate the function call into an assignment statement:

```
int x = foo(x)
```

The *main* Function

Every program requires at least one function, called *'main'*. This is where the program begins executing, and serves as its function definition.

Example:

```
func main()->int:
      // body
```

The main function may or may not return a value - execution of the program ends once the end of the main function is reached.

Recursive Functions

You can write a function that is recursive—a function that calls itself.

```
func factorial (int x):
      if (x < 1):
            return 1
      else:
            return (x * factorial (x-1))
```

Function declarations

All functions must be defined in the global scope - we do not allow nesting of function or actor declarations.

Program Structure and Scope

Program Structure

The only requirement to run your program is the main function. That will be the entry point for every program. Other than that, files can be linked to manually or by putting them in a central directory that will be specified when compiling.

Scope

Scope refers to what sections of the code the program can "see." All identifiers need to be defined before their usage.

Example 1:

```
// This is correct
actor_name():
      receive:
            do_anything():
                  print("hello")
main():
      actor new_actor = new actor_name()
      string response = actor_name.do_anything | new_actor
```

Example 2 (usage before definition):

```
// Error: actor_name is not defined at the time of usage
main():
      actor new_actor = new actor_name()
string response = actor_name.do_anything | new_actor

actor_name():
            receive:
                  do_anything():
                        print("hello")
```

Scope in Actors

Declarations made within actors are visible only within those actors. The initialization values of an actor is available across different sections (receive(), drop()) and message types of the actor. These are called actor-level variables. Inputs of messages are known as message-level variables. These message-level parameters are only available within the message body and go out of scope when the message body ends. New variables defined in receive() functions can shadow or overload existing actor local variables.

Scope in Functions

Declarations made within functions are visible only within those functions.

A Sample Program

The Hello World Example:

```
func main() -> int:
      print(“hello_world!”);
      return 0;
```

The Dolphin Example:

```
dolphin(int weight, string name):
      int doubleWeight = weight * 2;
      receive:
            eat(int num):
                  weight = weight + num
            follow_me(actor sender):
                  trainer.be_followed | sender
            catch(actor target):
                  fish.be_caught_and_eaten | target
            twin_yourself(int weight, string name):
                  actor twin = new dolphin(weight, name)
            be_free():
                  print(“So long, and thanks for all the fish!”)
      drop:
            print(“This is not allowed for a dolphin”)


trainer(string name):
      receive:
            be_followed():
                  print(“being followed!”)
            throw_fish_to_dolphin(actor target, actor food):
                  dolphin.catch(food) | target
      drop:
            print(“This is not allowed for a trainer”)

fish(int weight):
      receive:
            Be_caught_and_eaten:
```

```
                eat(weight) | sender
                // do not recursively call self, so the fish has died
        drop:
                print("I'm only a fish, I can't do this")
                fish()

/* This is the entry point for the program */
func main() -> int:
        actor bottlenose = new dolphin(3, "bottlenose")
        actor bob = new trainer("Bob")
        actor salmon = new fish(15)
        actor yellowtail = new fish(12)
        actor bluefin_tuna = new fish(100)
        trainer.throw_fish_to_dolphin(bottlenose, salmon) | bob
        trainer.throw_fish_to_dolphin(bottlenose, yellowtail) | bob
        trainer.throw_fish_to_dolphin(bottlenose, bluefin_tuna) | bob
        dolphin.be_free() | bottlenose
        return 0
```

References for our reference

We borrowed the structure of our LRM heavily from the C Language Reference Manual: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

Project Plan

Planning Process

Our team met to discuss the project about twice a week towards the beginning, and then more towards the end. Often these "meetings" were over Google Hangouts. For about the first half of the semester we met with our T.A. Danny Echikson every week or every other week. We would usually stay after to discuss further as a team. However, most of our time spent coding as a team or in pairs were over Google Hangouts using screen sharing.

In our meetings with Danny, he gave us guidance on our plan to implement an Actor model language. He told us that a previous team also did an Actor model language but that they imported libraries rather than using Ocaml LLVM library to implement many of the trademark actor features, and Danny challenged us to not rely on C imports and to implement those aspects ourselves in codegen, and that was a big influence on our planning. We are grateful we knew what a challenge that would be going in so that we could budget resources accordingly.

Our roadmap was mainly formed in brainstorming type conversations as a group where we would whiteboard out our vision for the language from a user perspective, and alo how we might implement. Later, when we were actually implementing things, if someone hit a roadblock we would usually be on a hangout with at least one other person, so we would then immediately talk out the problem, showing it to the other person, and come up with a plan for a solution or a new approach. This type of communication helped keep us all out of rabbit holes.

We were also very in touch with our big picture goals throughout. We kept track of constantly updated our priorities just using text docs but that was effective enough for us to keep us on track with our desired plan.

Programming Style Guide

Several rules that we tried to coordinate:

- Use underscores as opposed to camelCase
- Lines over 80 characters should be made to 2 separate lines
- Use spaces instead of tabs for indentation

For version control, we used Git, and our process convention was to work in independent branches and make pull requests to master when we're ready to add to the project - generally only pull request with all tests passing. Frequently rebase with master to incrementally deal with merge conflicts.

Show your project timeline

February 8th	Submitted project proposal
February 22nd	Submitted language reference manual
February 26th	Initial commit
March 25th	Hello world working
April 1st	Preprocessor done
April 23rd	Parser and scanner finalized by this point, actors added with minimal code generation
May 9th	Actor code generation complete
May 10th	Final project presentation

Please see the git log listing in the appendix for a full timeline of our project.

Identify roles and responsibilities of each team member

At first, we only had one role really defined (Betsy as manager, as a manager is required), and then also an idea of what people were more drawn to as such:

> Suraj: preprocessing, syntax and parsing, the ast, and our language's grammar overall; as well as building external C functions for our LLVM to call
> Linda: code generation and owner of LLVM;
> Mike: language's architectural design; actor implementation using C pthreads, code generation and LLVM
> Betsy: language design and "product management"--trying to focus on the big picture, make sure we were all avoiding rabbit holes, and that we were adhering to our Actor model goals.

However, as we progressed roles became both more fluid and, at times, more specialized. We are all really proud of how we worked together. We all were always ready and willing to determine together what was the next highest priority, and then individually or in subgroups, we would gladly attack what we decided was next in the queue.

Our Tech Stack

● Github- Hosted Git Repository - for version control
● Python 2.7 -.10 for a preprocessor that parses according to Theatr's whitespace scoping and also adds braces and semicolons (which are optional for the programmer)
● OCaml 4.2.01 - for parsing and semantic checking of our preprocessed Theatr code and the generation of LLVM via the ocaml API for our backend language logic in codegen
● LLVM IR as the immediate target for our compiler
● LLC to convert our LLVM IR to assembly
● GCC- for linking external C libraries, compiling our own C libraries, and building the executable binary

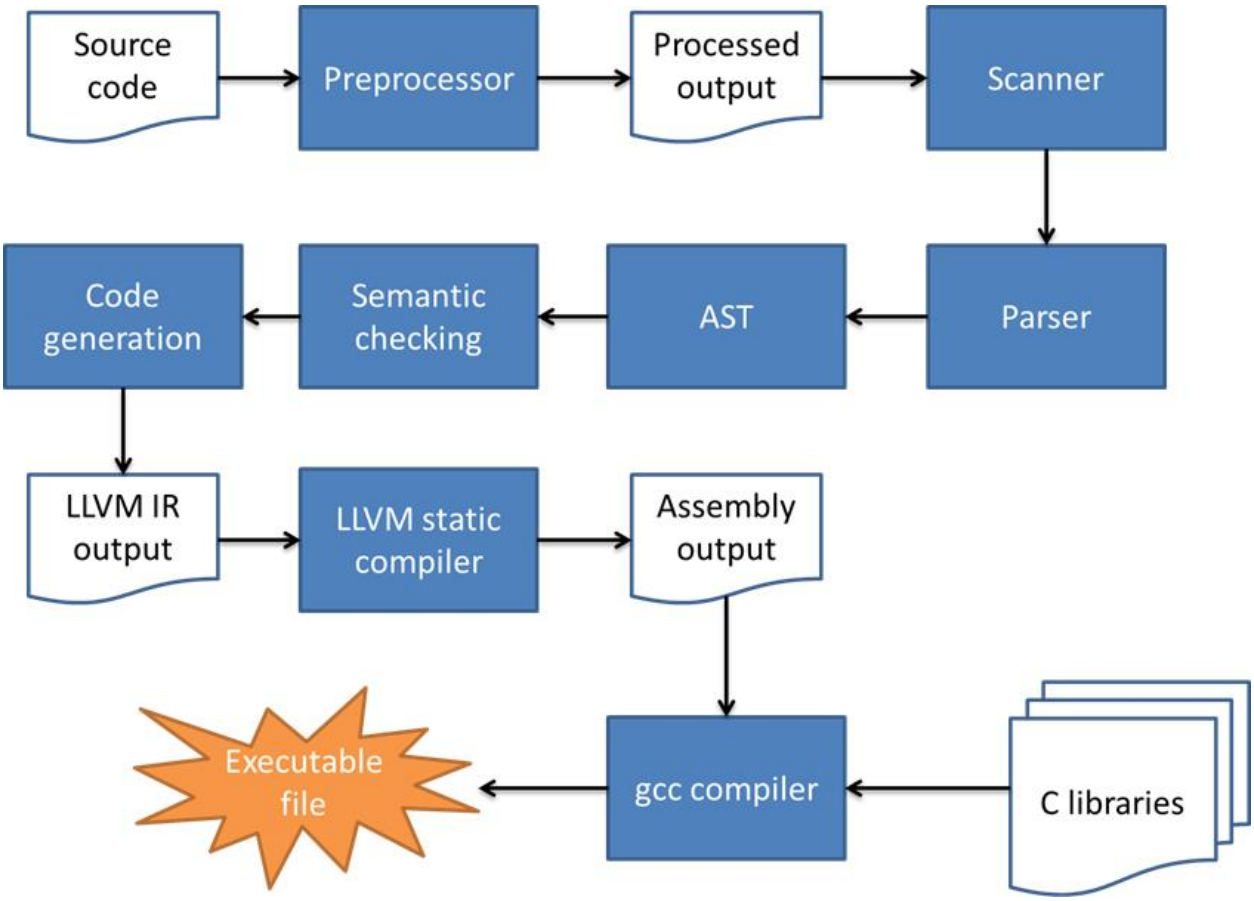
Project Log

(See end of Appendix for Git log.)
Theater's git history shows a log of 221 commits between Feb 26th and May 10th 2017. As is clear from the commit history, each member worked consistently and throughout. However,

because we paired so much, there was a lot of activity that is not tracked by this commit history in terms of overlapping programmers.

Architectural Design

Block diagram



Preprocessor - preprocessor.py

Theatr's preprocessor is written in Python and takes in the source code written by the user in a *.th file in a format which is very similar to python, and outputs a *.temp file that can be read by the scanner, by replacing line indents (denoted by 4 spaces per indent) with braces and end of line with semicolons, if one is not present. An example of source code:

```
dolphin(int weight):
    receive:
        eat(int food):
```

```
                print("Dolphin: thanks for the fish!")
                weight = weight + food
        drop:
            print("I couldn't understand that.")

    func main() -> int:
        actor emerson = new dolphin(12)
        dolphin.eat(5) | emerson
        dolphin.die() | emerson
        return 0
```

Converted into postprocessed code:

```
    dolphin(int weight):{
        receive:{
            eat(int food):{
                print("Dolphin: thanks for the fish!");
                weight = weight + food;}}
        drop:{
            print("I couldn't understand that.");}}
    func main() -> int:{
        actor emerson = new dolphin(12);
        dolphin.eat(5) | emerson;
        dolphin.die() | emerson;
        return 0;}
```

The processor starts with removing all the comments from the source code in .th format. It replaces all the whitespace character between new lines with a new line. We use a stack based method to keep track of tabs and newline characters. Each entry in the stack is (a = new_line_character_position, b = number of tabs following new line). We Iterate through the stack in reverse order. The stack always has bigger blocks above smaller block. The order is defined by b. We use this stack to replace enclose tab indented section of code with braces.

Scanner, parser, and AST: scanner.mll, parser.mly, ast.ml

The scanner, written in ocamllex, takes in output from the preprocessor and tokenizes it into keywords, ids, and literals. String literals can contain escape characters to express special characters. At this step, whitespace is removed as it is no longer needed to separate tokens. The parser is written in ocamlyacc, defines a grammar, and takes a series of tokens passed in from the scanner along with definitions provided by ast.ml to generate an AST. After these steps, the scanner and parser ensure that the program is syntactically (but not necessarily semantically) correct.

Semantic checking and code generation: semant.ml, codegen.ml

The compilation makes a single pass through the AST, so both the semantic checker and code generation are running simultaneously. As part of code generation, StringMaps are used to keep track of global and local variables as they are declared, to ensure there are no duplicates. Our codegen.ml is written in ocaml using the ocaml API for LLVM, and the output of this step is a program compiled down to LLVM IR.

C libraries: queue.c/h, filedwld.c/h, C standard libaries

We use the queue data structure for message queue of each actor, which is our own implementation in C, which uses the C library mutex and condition variables. We also provide file downloading functionality in filedwld.c/h. These .c files get compiled to .o files using gcc, which are linked and called by our LLVM IR.

Queue:

```
typedef struct message {
    int val; /* this is the match case*/
    /*this struct should hold the arguments for the function to be executed*/
    void *argumentStruct;
    /* a pointer to the queue of the sender */
    void *sender;
} message_t;
```

The message format is shown above. Each of these messages are enclosed in a queue entry structure as shown below.

```
typedef struct queue {
   struct queue *next;
   message_t message;
} queue_t;
```

Since messages are added/removed from each queue asynchronously by threads, we need a mutex lock (spin lock could also be used) to avoid race conditions. Also, we want the actor associated with a queue to wait for a message when the queue is empty. We use a condition variable and a count variable to facilitate this behavior. To hold a lock, a condition variable, and a count (of number of entries in the queue) for each queue, we define a separate structure called head which also contains the pointer to the back of the queue.

```
typedef struct head {
   queue_t *queue;
   int count;
   pthread_cond_t count_cond;
   pthread_mutex_t lock;
} head;
```

```
head *initialize_queue();
void enqueue(head *qhead, message_t message);
message_t dequeue(head *qhead);
```

We expose an enqueue, a dequeue, and a initialization function to the user. The initialize_queue function can be used to get the head of the queue with initialized mutex lock, condition variable, count set to zero, and a null pointer to the queue.

File Download

We add a C function to download file to our standard library. The interface is shown below.

```
/* httpa is a string of url to download
   filename is a string of filename where the downloaded data would be saved
*/
int geturl(char *httpa, char *filename)
```

LLVM static compiler and gcc compiler

We use gcc to link our compiled C code and the C libraries to our LLVM IR. In order to do this, we convert the LLVM IR into a .s assembler file using LLC. gcc then links the .s file with the C libraries and outputs an executable binary, of the type .exe.

Test Plan

Web crawler: demo-actor-download

Below is a sample program that spins up 3 actors to download a list of files:

```
crawler():
    receive:
        dwld(string url, string filename):
            geturl(url, filename);
            print("File downloaded!!");
        myprint(string name):
            print(name);
    drop:
        print("not getting paid well enough to do that job")
    after:
        return

func main() -> int:
    actor c1 = new crawler()
    actor c2 = new crawler()
    actor c3 = new crawler()
    string url1 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/intro.pdf"
    string url2 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/processors.pdf"
    string url3 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/projects.pdf"
    string url4 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/syntax.pdf"
    string url5 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/hw1.pdf"
    string url6 = "http://www.cs.columbia.edu/~sedwards/classes/2017/4115-
spring/microc.pdf"
    crawler.dwld(url1, "intro.pdf") | c1
    crawler.dwld(url2, "processor.pdf") | c2
    crawler.dwld(url3, "projects.pdf") | c3
    crawler.myprint("crawler 1 downloaded intro.pdf") | c1
    crawler.myprint("crawler 2 downloaded processor.pdf") | c2
    crawler.myprint("crawler 3 downloaded projects.pdf") | c3
    crawler.gfy() | c1
    crawler.dwld(url1, "syntax.pdf") | c1
    crawler.dwld(url1, "hw1.pdf") | c2
    crawler.dwld(url1, "microc.pdf") | c3
    crawler.myprint("crawler 1 downloaded syntax.pdf") | c1
    crawler.myprint("crawler 2 downloaded hw1.pdf") | c2
    crawler.myprint("crawler 3 downloaded microc.pdf") | c3
    crawler.die() | c1
    crawler.die() | c2
    crawler.die() | c3
    return 0;
```

The LLVM generated for the demo-actor-download.th is:

```
; ModuleID = 'Theatr'

%actor_address_struct = type { i32, i32, i8* }
%struct.head = type { %struct.queue*, %union.pthread_cond_t, %union.pthread_mutex_t }
%struct.queue = type { %struct.queue*, %struct.message }
%struct.message = type { i32, i8*, i8* }
%union.pthread_cond_t = type { %struct.anon, [4 x i8*] }
%struct.anon = type { i32, i32, i64, i64, i64, i8*, i32, i32 }
%union.pthread_mutex_t = type { %struct.__pthread_mutex_s }
%struct.__pthread_mutex_s = type { i32, i32, i32, i32, i32, %union.anon }
%union.anon = type { %struct.__pthread_internal_slist }
%struct.__pthread_internal_slist = type { %struct.__pthread_internal_slist* }
%crawler_struct = type { i32 }
%temp = type { i8*, i8* }
%temp.0 = type { i8*, i8* }
%temp.1 = type { i8*, i8* }
%temp.2 = type { i8* }
%temp.3 = type { i8* }
%temp.4 = type { i8* }
%temp.5 = type {}
%temp.6 = type { i8*, i8* }
%temp.7 = type { i8*, i8* }
%temp.8 = type { i8*, i8* }
%temp.9 = type { i8* }
%temp.10 = type { i8* }
%temp.11 = type { i8* }
%temp.12 = type {}
%temp.13 = type {}
%temp.14 = type {}
%dwld_struct = type { i8*, i8* }
%myprint_struct = type { i8* }

@global_actors = global [1024 x %actor_address_struct] zeroinitializer
@actor_count = global i32 1
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt3 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str = private unnamed_addr constant [44 x i8] c"not getting paid well enough to do that job\00"
@fmt4 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt5 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt6 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt7 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str8 = private unnamed_addr constant [18 x i8] c"File downloaded!!\00"
@fmt9 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
```

```
@fmt10 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt11 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt12 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str13 = private unnamed_addr constant [72 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/intro.pdf\00"
@.str14 = private unnamed_addr constant [77 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/processors.pdf\00"
@.str15 = private unnamed_addr constant [75 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/projects.pdf\00"
@.str16 = private unnamed_addr constant [73 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/syntax.pdf\00"
@.str17 = private unnamed_addr constant [70 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/hw1.pdf\00"
@.str18 = private unnamed_addr constant [73 x i8]
c"http://www.cs.columbia.edu/~sedwards/classes/2017/4115-spring/microc.pdf\00"
@.str19 = private unnamed_addr constant [10 x i8] c"intro.pdf\00"
@.str20 = private unnamed_addr constant [14 x i8] c"processor.pdf\00"
@.str21 = private unnamed_addr constant [13 x i8] c"projects.pdf\00"
@.str22 = private unnamed_addr constant [31 x i8] c"crawler 1 downloaded intro.pdf\00"
@.str23 = private unnamed_addr constant [35 x i8] c"crawler 2 downloaded processor.pdf\00"
@.str24 = private unnamed_addr constant [34 x i8] c"crawler 3 downloaded projects.pdf\00"
@.str25 = private unnamed_addr constant [11 x i8] c"syntax.pdf\00"
@.str26 = private unnamed_addr constant [8 x i8] c"hw1.pdf\00"
@.str27 = private unnamed_addr constant [11 x i8] c"microc.pdf\00"
@.str28 = private unnamed_addr constant [32 x i8] c"crawler 1 downloaded syntax.pdf\00"
@.str29 = private unnamed_addr constant [29 x i8] c"crawler 2 downloaded hw1.pdf\00"
@.str30 = private unnamed_addr constant [32 x i8] c"crawler 3 downloaded microc.pdf\00"

declare %struct.head* @initialize_queue()

declare void @enqueue(%struct.head*, %struct.message)

declare void @dequeue(%struct.message*, %struct.head*)

declare i32 @geturl(i8*, i8*)

declare i32 @printf(i8*, ...)

declare i32 @pthread_create(i32*, i8*, i8* (i8*)*, i8*)

declare i32 @pthread_join(i32, i8**)

define i32 @main() {
entry:
  %c1 = alloca %actor_address_struct*
  %c2 = alloca %actor_address_struct*
```

```
  %c3 = alloca %actor_address_struct*
  %url1 = alloca i8*
  %url2 = alloca i8*
  %url3 = alloca i8*
  %url4 = alloca i8*
  %url5 = alloca i8*
  %url6 = alloca i8*
  %curr_actor_count = load i32* @actor_count
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count
  %0 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 0
  store i32 1, i32* %0
  %1 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 1
  %2 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %3 = call %struct.head* @initialize_queue()
  %4 = bitcast %struct.head* %3 to i8*
  store i8* %4, i8** %2
  %malloccall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32* null, i32 1) to i32))
  %5 = bitcast i8* %malloccall to %crawler_struct*
  %6 = getelementptr inbounds %crawler_struct* %5, i32 0, i32 0
  store i32 %curr_actor_count, i32* %6
  %7 = bitcast %crawler_struct* %5 to i8*
  %tid = alloca i32
  %pthread_create_result = call i32 @pthread_create(i32* %tid, i8* null, i8* (i8*)* @crawler, i8*
%7)
  %8 = load i32* %tid
  store i32 %8, i32* %1
  %9 = add i32 %curr_actor_count, 1
  store i32 %9, i32* @actor_count
  store %actor_address_struct* %pos, %actor_address_struct** %c1
  %curr_actor_count1 = load i32* @actor_count
  %pos2 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count1
  %10 = getelementptr inbounds %actor_address_struct* %pos2, i32 0, i32 0
  store i32 1, i32* %10
  %11 = getelementptr inbounds %actor_address_struct* %pos2, i32 0, i32 1
  %12 = getelementptr inbounds %actor_address_struct* %pos2, i32 0, i32 2
  %13 = call %struct.head* @initialize_queue()
  %14 = bitcast %struct.head* %13 to i8*
  store i8* %14, i8** %12
  %malloccall3 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32* null, i32 1) to i32))
  %15 = bitcast i8* %malloccall3 to %crawler_struct*
  %16 = getelementptr inbounds %crawler_struct* %15, i32 0, i32 0
  store i32 %curr_actor_count1, i32* %16
  %17 = bitcast %crawler_struct* %15 to i8*
  %tid4 = alloca i32
```

```
  %pthread_create_result5 = call i32 @pthread_create(i32* %tid4, i8* null, i8* (i8*)* @crawler, i8*
%17)
  %18 = load i32* %tid4
  store i32 %18, i32* %11
  %19 = add i32 %curr_actor_count1, 1
  store i32 %19, i32* @actor_count
  store %actor_address_struct* %pos2, %actor_address_struct** %c2
  %curr_actor_count6 = load i32* @actor_count
  %pos7 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count6
  %20 = getelementptr inbounds %actor_address_struct* %pos7, i32 0, i32 0
  store i32 1, i32* %20
  %21 = getelementptr inbounds %actor_address_struct* %pos7, i32 0, i32 1
  %22 = getelementptr inbounds %actor_address_struct* %pos7, i32 0, i32 2
  %23 = call %struct.head* @initialize_queue()
  %24 = bitcast %struct.head* %23 to i8*
  store i8* %24, i8** %22
  %malloccall8 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32* null, i32 1) to i32))
  %25 = bitcast i8* %malloccall8 to %crawler_struct*
  %26 = getelementptr inbounds %crawler_struct* %25, i32 0, i32 0
  store i32 %curr_actor_count6, i32* %26
  %27 = bitcast %crawler_struct* %25 to i8*
  %tid9 = alloca i32
  %pthread_create_result10 = call i32 @pthread_create(i32* %tid9, i8* null, i8* (i8*)* @crawler,
i8* %27)
  %28 = load i32* %tid9
  store i32 %28, i32* %21
  %29 = add i32 %curr_actor_count6, 1
  store i32 %29, i32* @actor_count
  store %actor_address_struct* %pos7, %actor_address_struct** %c3
  store i8* getelementptr inbounds ([72 x i8]* @.str13, i32 0, i32 0), i8** %url1
  store i8* getelementptr inbounds ([77 x i8]* @.str14, i32 0, i32 0), i8** %url2
  store i8* getelementptr inbounds ([75 x i8]* @.str15, i32 0, i32 0), i8** %url3
  store i8* getelementptr inbounds ([73 x i8]* @.str16, i32 0, i32 0), i8** %url4
  store i8* getelementptr inbounds ([70 x i8]* @.str17, i32 0, i32 0), i8** %url5
  store i8* getelementptr inbounds ([73 x i8]* @.str18, i32 0, i32 0), i8** %url6
  %addr_struct = load %actor_address_struct** %c1
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %30 = bitcast i8* %msgQueue to %struct.head*
  %31 = alloca %struct.message
  %url111 = load i8** %url1
  %malloccall12 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct = bitcast i8* %malloccall12 to %temp*
  %32 = alloca i8*
```

```
  store i8* %url111, i8** %32
  %arg_val = load i8** %32
  %33 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 0
  store i8* %arg_val, i8** %33
  %34 = alloca i8*
  store i8* getelementptr inbounds ([10 x i8]* @.str19, i32 0, i32 0), i8** %34
  %arg_val13 = load i8** %34
  %35 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 1
  store i8* %arg_val13, i8** %35
  %36 = bitcast %temp* %argument_struct to i8*
  %case = alloca i32
  store i32 1, i32* %case
  %37 = getelementptr inbounds %struct.message* %31, i32 0, i32 0
  %38 = getelementptr inbounds %struct.message* %31, i32 0, i32 1
  %39 = getelementptr inbounds %struct.message* %31, i32 0, i32 2
  store i32 1, i32* %37
  store i8* %36, i8** %38
  store i8* null, i8** %39
  %message_val = load %struct.message* %31
  call void @enqueue(%struct.head* %30, %struct.message %message_val)
  %addr_struct14 = load %actor_address_struct** %c2
  %msgQueue_ptr15 = getelementptr inbounds %actor_address_struct* %addr_struct14, i32 0, i32 2
  %msgQueue16 = load i8** %msgQueue_ptr15
  %40 = bitcast i8* %msgQueue16 to %struct.head*
  %41 = alloca %struct.message
  %url217 = load i8** %url2
  %malloccall18 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct19 = bitcast i8* %malloccall18 to %temp.0*
  %42 = alloca i8*
  store i8* %url217, i8** %42
  %arg_val20 = load i8** %42
  %43 = getelementptr inbounds %temp.0* %argument_struct19, i32 0, i32 0
  store i8* %arg_val20, i8** %43
  %44 = alloca i8*
  store i8* getelementptr inbounds ([14 x i8]* @.str20, i32 0, i32 0), i8** %44
  %arg_val21 = load i8** %44
  %45 = getelementptr inbounds %temp.0* %argument_struct19, i32 0, i32 1
  store i8* %arg_val21, i8** %45
  %46 = bitcast %temp.0* %argument_struct19 to i8*
  %case22 = alloca i32
  store i32 1, i32* %case22
  %47 = getelementptr inbounds %struct.message* %41, i32 0, i32 0
  %48 = getelementptr inbounds %struct.message* %41, i32 0, i32 1
  %49 = getelementptr inbounds %struct.message* %41, i32 0, i32 2
  store i32 1, i32* %47
```

```
  store i8* %46, i8** %48
  store i8* null, i8** %49
  %message_val23 = load %struct.message* %41
  call void @enqueue(%struct.head* %40, %struct.message %message_val23)
  %addr_struct24 = load %actor_address_struct** %c3
  %msgQueue_ptr25 = getelementptr inbounds %actor_address_struct* %addr_struct24, i32 0, i32 2
  %msgQueue26 = load i8** %msgQueue_ptr25
  %50 = bitcast i8* %msgQueue26 to %struct.head*
  %51 = alloca %struct.message
  %url327 = load i8** %url3
  %malloccall28 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct29 = bitcast i8* %malloccall28 to %temp.1*
  %52 = alloca i8*
  store i8* %url327, i8** %52
  %arg_val30 = load i8** %52
  %53 = getelementptr inbounds %temp.1* %argument_struct29, i32 0, i32 0
  store i8* %arg_val30, i8** %53
  %54 = alloca i8*
  store i8* getelementptr inbounds ([13 x i8]* @.str21, i32 0, i32 0), i8** %54
  %arg_val31 = load i8** %54
  %55 = getelementptr inbounds %temp.1* %argument_struct29, i32 0, i32 1
  store i8* %arg_val31, i8** %55
  %56 = bitcast %temp.1* %argument_struct29 to i8*
  %case32 = alloca i32
  store i32 1, i32* %case32
  %57 = getelementptr inbounds %struct.message* %51, i32 0, i32 0
  %58 = getelementptr inbounds %struct.message* %51, i32 0, i32 1
  %59 = getelementptr inbounds %struct.message* %51, i32 0, i32 2
  store i32 1, i32* %57
  store i8* %56, i8** %58
  store i8* null, i8** %59
  %message_val33 = load %struct.message* %51
  call void @enqueue(%struct.head* %50, %struct.message %message_val33)
  %addr_struct34 = load %actor_address_struct** %c1
  %msgQueue_ptr35 = getelementptr inbounds %actor_address_struct* %addr_struct34, i32 0, i32 2
  %msgQueue36 = load i8** %msgQueue_ptr35
  %60 = bitcast i8* %msgQueue36 to %struct.head*
  %61 = alloca %struct.message
  %malloccall37 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct38 = bitcast i8* %malloccall37 to %temp.2*
  %62 = alloca i8*
  store i8* getelementptr inbounds ([31 x i8]* @.str22, i32 0, i32 0), i8** %62
  %arg_val39 = load i8** %62
  %63 = getelementptr inbounds %temp.2* %argument_struct38, i32 0, i32 0
```

```
  store i8* %arg_val39, i8** %63
  %64 = bitcast %temp.2* %argument_struct38 to i8*
  %case40 = alloca i32
  store i32 2, i32* %case40
  %65 = getelementptr inbounds %struct.message* %61, i32 0, i32 0
  %66 = getelementptr inbounds %struct.message* %61, i32 0, i32 1
  %67 = getelementptr inbounds %struct.message* %61, i32 0, i32 2
  store i32 2, i32* %65
  store i8* %64, i8** %66
  store i8* null, i8** %67
  %message_val41 = load %struct.message* %61
  call void @enqueue(%struct.head* %60, %struct.message %message_val41)
  %addr_struct42 = load %actor_address_struct** %c2
  %msgQueue_ptr43 = getelementptr inbounds %actor_address_struct* %addr_struct42, i32 0, i32 2
  %msgQueue44 = load i8** %msgQueue_ptr43
  %68 = bitcast i8* %msgQueue44 to %struct.head*
  %69 = alloca %struct.message
  %malloccall45 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct46 = bitcast i8* %malloccall45 to %temp.3*
  %70 = alloca i8*
  store i8* getelementptr inbounds ([35 x i8]* @.str23, i32 0, i32 0), i8** %70
  %arg_val47 = load i8** %70
  %71 = getelementptr inbounds %temp.3* %argument_struct46, i32 0, i32 0
  store i8* %arg_val47, i8** %71
  %72 = bitcast %temp.3* %argument_struct46 to i8*
  %case48 = alloca i32
  store i32 2, i32* %case48
  %73 = getelementptr inbounds %struct.message* %69, i32 0, i32 0
  %74 = getelementptr inbounds %struct.message* %69, i32 0, i32 1
  %75 = getelementptr inbounds %struct.message* %69, i32 0, i32 2
  store i32 2, i32* %73
  store i8* %72, i8** %74
  store i8* null, i8** %75
  %message_val49 = load %struct.message* %69
  call void @enqueue(%struct.head* %68, %struct.message %message_val49)
  %addr_struct50 = load %actor_address_struct** %c3
  %msgQueue_ptr51 = getelementptr inbounds %actor_address_struct* %addr_struct50, i32 0, i32 2
  %msgQueue52 = load i8** %msgQueue_ptr51
  %76 = bitcast i8* %msgQueue52 to %struct.head*
  %77 = alloca %struct.message
  %malloccall53 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct54 = bitcast i8* %malloccall53 to %temp.4*
  %78 = alloca i8*
  store i8* getelementptr inbounds ([34 x i8]* @.str24, i32 0, i32 0), i8** %78
```

```
  %arg_val55 = load i8** %78
  %79 = getelementptr inbounds %temp.4* %argument_struct54, i32 0, i32 0
  store i8* %arg_val55, i8** %79
  %80 = bitcast %temp.4* %argument_struct54 to i8*
  %case56 = alloca i32
  store i32 2, i32* %case56
  %81 = getelementptr inbounds %struct.message* %77, i32 0, i32 0
  %82 = getelementptr inbounds %struct.message* %77, i32 0, i32 1
  %83 = getelementptr inbounds %struct.message* %77, i32 0, i32 2
  store i32 2, i32* %81
  store i8* %80, i8** %82
  store i8* null, i8** %83
  %message_val57 = load %struct.message* %77
  call void @enqueue(%struct.head* %76, %struct.message %message_val57)
  %addr_struct58 = load %actor_address_struct** %c1
  %msgQueue_ptr59 = getelementptr inbounds %actor_address_struct* %addr_struct58, i32 0, i32 2
  %msgQueue60 = load i8** %msgQueue_ptr59
  %84 = bitcast i8* %msgQueue60 to %struct.head*
  %85 = alloca %struct.message
  %malloccall61 = tail call i8* @malloc(i32 0)
  %argument_struct62 = bitcast i8* %malloccall61 to %temp.5*
  %case63 = alloca i32
  store i32 -1, i32* %case63
  %86 = getelementptr inbounds %struct.message* %85, i32 0, i32 0
  %87 = getelementptr inbounds %struct.message* %85, i32 0, i32 1
  %88 = getelementptr inbounds %struct.message* %85, i32 0, i32 2
  store i32 -1, i32* %86
  store i8* null, i8** %87
  store i8* null, i8** %88
  %message_val64 = load %struct.message* %85
  call void @enqueue(%struct.head* %84, %struct.message %message_val64)
  %addr_struct65 = load %actor_address_struct** %c1
  %msgQueue_ptr66 = getelementptr inbounds %actor_address_struct* %addr_struct65, i32 0, i32 2
  %msgQueue67 = load i8** %msgQueue_ptr66
  %89 = bitcast i8* %msgQueue67 to %struct.head*
  %90 = alloca %struct.message
  %url168 = load i8** %url1
  %malloccall69 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct70 = bitcast i8* %malloccall69 to %temp.6*
  %91 = alloca i8*
  store i8* %url168, i8** %91
  %arg_val71 = load i8** %91
  %92 = getelementptr inbounds %temp.6* %argument_struct70, i32 0, i32 0
  store i8* %arg_val71, i8** %92
  %93 = alloca i8*
```

```
  store i8* getelementptr inbounds ([11 x i8]* @.str25, i32 0, i32 0), i8** %93
  %arg_val72 = load i8** %93
  %94 = getelementptr inbounds %temp.6* %argument_struct70, i32 0, i32 1
  store i8* %arg_val72, i8** %94
  %95 = bitcast %temp.6* %argument_struct70 to i8*
  %case73 = alloca i32
  store i32 1, i32* %case73
  %96 = getelementptr inbounds %struct.message* %90, i32 0, i32 0
  %97 = getelementptr inbounds %struct.message* %90, i32 0, i32 1
  %98 = getelementptr inbounds %struct.message* %90, i32 0, i32 2
  store i32 1, i32* %96
  store i8* %95, i8** %97
  store i8* null, i8** %98
  %message_val74 = load %struct.message* %90
  call void @enqueue(%struct.head* %89, %struct.message %message_val74)
  %addr_struct75 = load %actor_address_struct** %c2
  %msgQueue_ptr76 = getelementptr inbounds %actor_address_struct* %addr_struct75, i32 0, i32 2
  %msgQueue77 = load i8** %msgQueue_ptr76
  %99 = bitcast i8* %msgQueue77 to %struct.head*
  %100 = alloca %struct.message
  %url178 = load i8** %url1
  %malloccall79 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct80 = bitcast i8* %malloccall79 to %temp.7*
  %101 = alloca i8*
  store i8* %url178, i8** %101
  %arg_val81 = load i8** %101
  %102 = getelementptr inbounds %temp.7* %argument_struct80, i32 0, i32 0
  store i8* %arg_val81, i8** %102
  %103 = alloca i8*
  store i8* getelementptr inbounds ([8 x i8]* @.str26, i32 0, i32 0), i8** %103
  %arg_val82 = load i8** %103
  %104 = getelementptr inbounds %temp.7* %argument_struct80, i32 0, i32 1
  store i8* %arg_val82, i8** %104
  %105 = bitcast %temp.7* %argument_struct80 to i8*
  %case83 = alloca i32
  store i32 1, i32* %case83
  %106 = getelementptr inbounds %struct.message* %100, i32 0, i32 0
  %107 = getelementptr inbounds %struct.message* %100, i32 0, i32 1
  %108 = getelementptr inbounds %struct.message* %100, i32 0, i32 2
  store i32 1, i32* %106
  store i8* %105, i8** %107
  store i8* null, i8** %108
  %message_val84 = load %struct.message* %100
  call void @enqueue(%struct.head* %99, %struct.message %message_val84)
  %addr_struct85 = load %actor_address_struct** %c3
```

```
  %msgQueue_ptr86 = getelementptr inbounds %actor_address_struct* %addr_struct85, i32 0, i32 2
  %msgQueue87 = load i8** %msgQueue_ptr86
  %109 = bitcast i8* %msgQueue87 to %struct.head*
  %110 = alloca %struct.message
  %url188 = load i8** %url1
  %malloccall89 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct90 = bitcast i8* %malloccall89 to %temp.8*
  %111 = alloca i8*
  store i8* %url188, i8** %111
  %arg_val91 = load i8** %111
  %112 = getelementptr inbounds %temp.8* %argument_struct90, i32 0, i32 0
  store i8* %arg_val91, i8** %112
  %113 = alloca i8*
  store i8* getelementptr inbounds ([11 x i8]* @.str27, i32 0, i32 0), i8** %113
  %arg_val92 = load i8** %113
  %114 = getelementptr inbounds %temp.8* %argument_struct90, i32 0, i32 1
  store i8* %arg_val92, i8** %114
  %115 = bitcast %temp.8* %argument_struct90 to i8*
  %case93 = alloca i32
  store i32 1, i32* %case93
  %116 = getelementptr inbounds %struct.message* %110, i32 0, i32 0
  %117 = getelementptr inbounds %struct.message* %110, i32 0, i32 1
  %118 = getelementptr inbounds %struct.message* %110, i32 0, i32 2
  store i32 1, i32* %116
  store i8* %115, i8** %117
  store i8* null, i8** %118
  %message_val94 = load %struct.message* %110
  call void @enqueue(%struct.head* %109, %struct.message %message_val94)
  %addr_struct95 = load %actor_address_struct** %c1
  %msgQueue_ptr96 = getelementptr inbounds %actor_address_struct* %addr_struct95, i32 0, i32 2
  %msgQueue97 = load i8** %msgQueue_ptr96
  %119 = bitcast i8* %msgQueue97 to %struct.head*
  %120 = alloca %struct.message
  %malloccall98 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct99 = bitcast i8* %malloccall98 to %temp.9*
  %121 = alloca i8*
  store i8* getelementptr inbounds ([32 x i8]* @.str28, i32 0, i32 0), i8** %121
  %arg_val100 = load i8** %121
  %122 = getelementptr inbounds %temp.9* %argument_struct99, i32 0, i32 0
  store i8* %arg_val100, i8** %122
  %123 = bitcast %temp.9* %argument_struct99 to i8*
  %case101 = alloca i32
  store i32 2, i32* %case101
  %124 = getelementptr inbounds %struct.message* %120, i32 0, i32 0
```

```
  %125 = getelementptr inbounds %struct.message* %120, i32 0, i32 1
  %126 = getelementptr inbounds %struct.message* %120, i32 0, i32 2
  store i32 2, i32* %124
  store i8* %123, i8** %125
  store i8* null, i8** %126
  %message_val102 = load %struct.message* %120
  call void @enqueue(%struct.head* %119, %struct.message %message_val102)
  %addr_struct103 = load %actor_address_struct** %c2
  %msgQueue_ptr104 = getelementptr inbounds %actor_address_struct* %addr_struct103, i32 0, i32 2
  %msgQueue105 = load i8** %msgQueue_ptr104
  %127 = bitcast i8* %msgQueue105 to %struct.head*
  %128 = alloca %struct.message
  %malloccall106 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct107 = bitcast i8* %malloccall106 to %temp.10*
  %129 = alloca i8*
  store i8* getelementptr inbounds ([29 x i8]* @.str29, i32 0, i32 0), i8** %129
  %arg_val108 = load i8** %129
  %130 = getelementptr inbounds %temp.10* %argument_struct107, i32 0, i32 0
  store i8* %arg_val108, i8** %130
  %131 = bitcast %temp.10* %argument_struct107 to i8*
  %case109 = alloca i32
  store i32 2, i32* %case109
  %132 = getelementptr inbounds %struct.message* %128, i32 0, i32 0
  %133 = getelementptr inbounds %struct.message* %128, i32 0, i32 1
  %134 = getelementptr inbounds %struct.message* %128, i32 0, i32 2
  store i32 2, i32* %132
  store i8* %131, i8** %133
  store i8* null, i8** %134
  %message_val110 = load %struct.message* %128
  call void @enqueue(%struct.head* %127, %struct.message %message_val110)
  %addr_struct111 = load %actor_address_struct** %c3
  %msgQueue_ptr112 = getelementptr inbounds %actor_address_struct* %addr_struct111, i32 0, i32 2
  %msgQueue113 = load i8** %msgQueue_ptr112
  %135 = bitcast i8* %msgQueue113 to %struct.head*
  %136 = alloca %struct.message
  %malloccall114 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to
i32))
  %argument_struct115 = bitcast i8* %malloccall114 to %temp.11*
  %137 = alloca i8*
  store i8* getelementptr inbounds ([32 x i8]* @.str30, i32 0, i32 0), i8** %137
  %arg_val116 = load i8** %137
  %138 = getelementptr inbounds %temp.11* %argument_struct115, i32 0, i32 0
  store i8* %arg_val116, i8** %138
  %139 = bitcast %temp.11* %argument_struct115 to i8*
  %case117 = alloca i32
```

```
store i32 2, i32* %case117
%140 = getelementptr inbounds %struct.message* %136, i32 0, i32 0
%141 = getelementptr inbounds %struct.message* %136, i32 0, i32 1
%142 = getelementptr inbounds %struct.message* %136, i32 0, i32 2
store i32 2, i32* %140
store i8* %139, i8** %141
store i8* null, i8** %142
%message_val118 = load %struct.message* %136
call void @enqueue(%struct.head* %135, %struct.message %message_val118)
%addr_struct119 = load %actor_address_struct** %c1
%msgQueue_ptr120 = getelementptr inbounds %actor_address_struct* %addr_struct119, i32 0, i32 2
%msgQueue121 = load i8** %msgQueue_ptr120
%143 = bitcast i8* %msgQueue121 to %struct.head*
%144 = alloca %struct.message
%malloccall122 = tail call i8* @malloc(i32 0)
%argument_struct123 = bitcast i8* %malloccall122 to %temp.12*
%case124 = alloca i32
store i32 0, i32* %case124
%145 = getelementptr inbounds %struct.message* %144, i32 0, i32 0
%146 = getelementptr inbounds %struct.message* %144, i32 0, i32 1
%147 = getelementptr inbounds %struct.message* %144, i32 0, i32 2
store i32 0, i32* %145
store i8* null, i8** %146
store i8* null, i8** %147
%message_val125 = load %struct.message* %144
call void @enqueue(%struct.head* %143, %struct.message %message_val125)
%addr_struct126 = load %actor_address_struct** %c2
%msgQueue_ptr127 = getelementptr inbounds %actor_address_struct* %addr_struct126, i32 0, i32 2
%msgQueue128 = load i8** %msgQueue_ptr127
%148 = bitcast i8* %msgQueue128 to %struct.head*
%149 = alloca %struct.message
%malloccall129 = tail call i8* @malloc(i32 0)
%argument_struct130 = bitcast i8* %malloccall129 to %temp.13*
%case131 = alloca i32
store i32 0, i32* %case131
%150 = getelementptr inbounds %struct.message* %149, i32 0, i32 0
%151 = getelementptr inbounds %struct.message* %149, i32 0, i32 1
%152 = getelementptr inbounds %struct.message* %149, i32 0, i32 2
store i32 0, i32* %150
store i8* null, i8** %151
store i8* null, i8** %152
%message_val132 = load %struct.message* %149
call void @enqueue(%struct.head* %148, %struct.message %message_val132)
%addr_struct133 = load %actor_address_struct** %c3
%msgQueue_ptr134 = getelementptr inbounds %actor_address_struct* %addr_struct133, i32 0, i32 2
%msgQueue135 = load i8** %msgQueue_ptr134
```

```
  %153 = bitcast i8* %msgQueue135 to %struct.head*
  %154 = alloca %struct.message
  %malloccall136 = tail call i8* @malloc(i32 0)
  %argument_struct137 = bitcast i8* %malloccall136 to %temp.14*
  %case138 = alloca i32
  store i32 0, i32* %case138
  %155 = getelementptr inbounds %struct.message* %154, i32 0, i32 0
  %156 = getelementptr inbounds %struct.message* %154, i32 0, i32 1
  %157 = getelementptr inbounds %struct.message* %154, i32 0, i32 2
  store i32 0, i32* %155
  store i8* null, i8** %156
  store i8* null, i8** %157
  %message_val139 = load %struct.message* %154
  call void @enqueue(%struct.head* %153, %struct.message %message_val139)
  %158 = alloca i32
  store i32 1, i32* %158
  br label %while

while:                                            ; preds = %while_body, %entry
  %"return:i141" = load i32* %158
  %tmp142 = icmp slt i32 %"return:i141", 1024
  br i1 %tmp142, label %while_body, label %merge

while_body:                                       ; preds = %while
  %idx = load i32* %158
  %pos140 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32 %idx
  %tid_p = getelementptr inbounds %actor_address_struct* %pos140, i32 0, i32 1
  %tid_val = load i32* %tid_p
  %pthread_join_result = call i32 @pthread_join(i32 %tid_val, i8** null)
  %"return:i" = load i32* %158
  %tmp = add i32 %"return:i", 1
  store i32 %tmp, i32* %158
  br label %while

merge:                                            ; preds = %while
  ret i32 0
}

define i8* @crawler(i8* %ptr) {
entry:
  %0 = alloca i8*
  store i8* %ptr, i8** %0
  %state_struct = alloca %crawler_struct*
  %1 = load i8** %0
  %2 = bitcast i8* %1 to %crawler_struct*
  store %crawler_struct* %2, %crawler_struct** %state_struct
```

```
  %3 = load %crawler_struct** %state_struct
  %4 = getelementptr inbounds %crawler_struct* %3, i32 0, i32 0
  %5 = load i32* %4
  %self_index = alloca i32
  store i32 %5, i32* %self_index
  tail call void @free(i8* %ptr)
  br label %pred_msg_while_bb

pred_msg_while_bb:                                ; preds = %entry, %finish_msg_while_bb
  br i1 true, label %body_msg_while_bb, label %merge_msg_while_bb

body_msg_while_bb:                                ; preds = %pred_msg_while_bb
  %"self:index_val" = load i32* %self_index
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%"self:index_val"
  %6 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %7 = load i8** %6
  %8 = bitcast i8* %7 to %struct.head*
  %message_struct = alloca %struct.message
  call void @dequeue(%struct.message* %message_struct, %struct.head* %8)
  %9 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 0
  %case_num = load i32* %9
  %10 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 1
  %actuals_ptr = load i8** %10
  %11 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 2
  %sender_ptr = load i8** %11
  switch i32 %case_num, label %msg_default_case_bb [
    i32 1, label %msg_dwld_case_bb
    i32 2, label %msg_myprint_case_bb
    i32 0, label %msg_die_case_bb
  ]

finish_msg_while_bb:                              ; preds = %msg_myprint_case_bb,
%msg_dwld_case_bb, %msg_default_case_bb
  br label %pred_msg_while_bb

merge_msg_while_bb:                               ; preds = %msg_die_case_bb, %pred_msg_while_bb
  %ret = alloca i8
  ret i8* %ret

msg_die_case_bb:                                  ; preds = %body_msg_while_bb
  br label %merge_msg_while_bb

msg_dwld_case_bb:                                 ; preds = %body_msg_while_bb
  %actual_ptr = bitcast i8* %actuals_ptr to %dwld_struct*
  %12 = alloca %dwld_struct*
```

```
  store %dwld_struct* %actual_ptr, %dwld_struct** %12
  %13 = load %dwld_struct** %12
  %f_ptr = getelementptr inbounds %dwld_struct* %13, i32 0, i32 0
  %14 = load i8** %f_ptr
  %url = alloca i8*
  store i8* %14, i8** %url
  %f_ptr1 = getelementptr inbounds %dwld_struct* %13, i32 0, i32 1
  %15 = load i8** %f_ptr1
  %filename = alloca i8*
  store i8* %15, i8** %filename
  %filename2 = load i8** %filename
  %url3 = load i8** %url
  %geturl = call i32 @geturl(i8* %url3, i8* %filename2)
  %printf4 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt6, i32 0, i32
0), i8* getelementptr inbounds ([18 x i8]* @.str8, i32 0, i32 0))
  br label %finish_msg_while_bb

msg_myprint_case_bb:                              ; preds = %body_msg_while_bb
  %actual_ptr5 = bitcast i8* %actuals_ptr to %myprint_struct*
  %16 = alloca %myprint_struct*
  store %myprint_struct* %actual_ptr5, %myprint_struct** %16
  %17 = load %myprint_struct** %16
  %f_ptr6 = getelementptr inbounds %myprint_struct* %17, i32 0, i32 0
  %18 = load i8** %f_ptr6
  %name = alloca i8*
  store i8* %18, i8** %name
  %name7 = load i8** %name
  %printf8 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt11, i32 0, i32
0), i8* %name7)
  br label %finish_msg_while_bb

msg_default_case_bb:                              ; preds = %body_msg_while_bb
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt2, i32 0, i32
0), i8* getelementptr inbounds ([44 x i8]* @.str, i32 0, i32 0))
  br label %finish_msg_while_bb
}

declare void @free(i8*)

declare noalias i8* @malloc(i32)
```

Actors creating new actors: test-new-actor-in-actor.th

The following program demonstrates creating new actors from within an actor and sending messages to it.

```
dolphin(int weight, int age):
    print("dolphin here")
    print(weight)
    print(age)
    actor baby = new babyDolphin(weight*2, age*2)
    babyDolphin.die() | baby
    receive:
        eat(int food):
            weight = weight + food
            print("dolphin eating, now weighs")
            print(weight)
        growOld(int time):
            age = age + time
        eatAndGrowOld(int food, int time):
            weight = weight + food
            age = age + time
    drop:
        print("inside drop()")

babyDolphin(int weight, int age):
    print("baby dolphin here")
    print(weight)
    print(age)
    receive:
        eat(int food):
            weight = weight + food
            print("baby dolphin eating, now weighs")
            print(weight)
    drop:
        print("inside drop()")

func main() -> int:
    int weight = 100
    int age = 3
    actor d = new dolphin(weight, age)
    dolphin.die() | d
    return 0
```

The LLVM generated for the test-new-actor-in-actor.th is:

```
; ModuleID = 'Theatr'

%actor_address_struct = type { i32, i32, i8* }
%struct.head = type { %struct.queue*, %union.pthread_cond_t, %union.pthread_mutex_t }
%struct.queue = type { %struct.queue*, %struct.message }
%struct.message = type { i32, i8*, i8* }
%union.pthread_cond_t = type { %struct.anon, [4 x i8*] }
%struct.anon = type { i32, i32, i64, i64, i64, i8*, i32, i32 }
%union.pthread_mutex_t = type { %struct.__pthread_mutex_s }
%struct.__pthread_mutex_s = type { i32, i32, i32, i32, i32, %union.anon }
%union.anon = type { %struct.__pthread_internal_slist }
%struct.__pthread_internal_slist = type { %struct.__pthread_internal_slist* }
%dolphin_struct = type { i32, i32, i32 }
%temp.1 = type {}
%babyDolphin_struct = type { i32, i32, i32 }
%eat_struct = type { i32 }
%temp = type {}
%eat_struct.0 = type { i32 }
%growOld_struct = type { i32 }
%eatAndGrowOld_struct = type { i32, i32 }

@global_actors = global [1024 x %actor_address_struct] zeroinitializer
@actor_count = global i32 1
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt3 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str = private unnamed_addr constant [18 x i8] c"baby dolphin here\00"
@fmt4 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt5 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt6 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt7 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@fmt8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt9 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt10 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt11 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@fmt12 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt13 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt14 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt15 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str16 = private unnamed_addr constant [14 x i8] c"inside drop()\00"
@fmt17 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt18 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt19 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt20 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
```

```
@.str21 = private unnamed_addr constant [32 x i8] c"baby dolphin eating, now weighs\00"
@fmt22 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt23 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt24 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt25 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@fmt26 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt27 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt28 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt29 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str30 = private unnamed_addr constant [13 x i8] c"dolphin here\00"
@fmt31 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt32 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt33 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt34 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@fmt35 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt36 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt37 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt38 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@fmt39 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt40 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt41 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt42 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str43 = private unnamed_addr constant [14 x i8] c"inside drop()\00"
@fmt44 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt45 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt46 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt47 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str48 = private unnamed_addr constant [27 x i8] c"dolphin eating, now weighs\00"
@fmt49 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt50 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt51 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt52 = private unnamed_addr constant [4 x i8] c"%c\0A\00"

declare %struct.head* @initialize_queue()

declare void @enqueue(%struct.head*, %struct.message)

declare void @dequeue(%struct.message*, %struct.head*)

declare i32 @geturl(i8*, i8*)

declare i32 @printf(i8*, ...)

declare i32 @pthread_create(i32*, i8*, i8* (i8*)*, i8*)

declare i32 @pthread_join(i32, i8**)
```

```
define i32 @main() {
entry:
  %weight = alloca i32
  %age = alloca i32
  %d = alloca %actor_address_struct*
  store i32 100, i32* %weight
  store i32 3, i32* %age
  %age1 = load i32* %age
  %weight2 = load i32* %weight
  %curr_actor_count = load i32* @actor_count
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count
  %0 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 0
  store i32 1, i32* %0
  %1 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 1
  %2 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %3 = call %struct.head* @initialize_queue()
  %4 = bitcast %struct.head* %3 to i8*
  store i8* %4, i8** %2
  %malloccall = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i32* getelementptr (i32*
null, i32 1) to i64), i64 3) to i32))
  %5 = bitcast i8* %malloccall to %dolphin_struct*
  %6 = getelementptr inbounds %dolphin_struct* %5, i32 0, i32 0
  store i32 %curr_actor_count, i32* %6
  %7 = getelementptr inbounds %dolphin_struct* %5, i32 0, i32 1
  store i32 %weight2, i32* %7
  %8 = getelementptr inbounds %dolphin_struct* %5, i32 0, i32 2
  store i32 %age1, i32* %8
  %9 = bitcast %dolphin_struct* %5 to i8*
  %tid = alloca i32
  %pthread_create_result = call i32 @pthread_create(i32* %tid, i8* null, i8* (i8*)* @dolphin, i8*
%9)
  %10 = load i32* %tid
  store i32 %10, i32* %1
  %11 = add i32 %curr_actor_count, 1
  store i32 %11, i32* @actor_count
  store %actor_address_struct* %pos, %actor_address_struct** %d
  %addr_struct = load %actor_address_struct** %d
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %12 = bitcast i8* %msgQueue to %struct.head*
  %13 = alloca %struct.message
  %malloccall3 = tail call i8* @malloc(i32 0)
  %argument_struct = bitcast i8* %malloccall3 to %temp.1*
  %case = alloca i32
```

```
  store i32 0, i32* %case
  %14 = getelementptr inbounds %struct.message* %13, i32 0, i32 0
  %15 = getelementptr inbounds %struct.message* %13, i32 0, i32 1
  %16 = getelementptr inbounds %struct.message* %13, i32 0, i32 2
  store i32 0, i32* %14
  store i8* null, i8** %15
  store i8* null, i8** %16
  %message_val = load %struct.message* %13
  call void @enqueue(%struct.head* %12, %struct.message %message_val)
  %17 = alloca i32
  store i32 1, i32* %17
  br label %while

while:                                            ; preds = %while_body, %entry
  %"return:i5" = load i32* %17
  %tmp6 = icmp slt i32 %"return:i5", 1024
  br i1 %tmp6, label %while_body, label %merge

while_body:                                       ; preds = %while
  %idx = load i32* %17
  %pos4 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32 %idx
  %tid_p = getelementptr inbounds %actor_address_struct* %pos4, i32 0, i32 1
  %tid_val = load i32* %tid_p
  %pthread_join_result = call i32 @pthread_join(i32 %tid_val, i8** null)
  %"return:i" = load i32* %17
  %tmp = add i32 %"return:i", 1
  store i32 %tmp, i32* %17
  br label %while

merge:                                            ; preds = %while
  ret i32 0
}

define i8* @babyDolphin(i8* %ptr) {
entry:
  %0 = alloca i8*
  store i8* %ptr, i8** %0
  %state_struct = alloca %babyDolphin_struct*
  %1 = load i8** %0
  %2 = bitcast i8* %1 to %babyDolphin_struct*
  store %babyDolphin_struct* %2, %babyDolphin_struct** %state_struct
  %3 = load %babyDolphin_struct** %state_struct
  %4 = getelementptr inbounds %babyDolphin_struct* %3, i32 0, i32 0
  %5 = load i32* %4
  %self_index = alloca i32
  store i32 %5, i32* %self_index
```

```
  %6 = load %babyDolphin_struct** %state_struct
  %7 = getelementptr inbounds %babyDolphin_struct* %6, i32 0, i32 1
  %8 = load i32* %7
  %weight = alloca i32
  store i32 %8, i32* %weight
  %9 = load %babyDolphin_struct** %state_struct
  %10 = getelementptr inbounds %babyDolphin_struct* %9, i32 0, i32 2
  %11 = load i32* %10
  %age = alloca i32
  store i32 %11, i32* %age
  tail call void @free(i8* %ptr)
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt2, i32 0, i32 0),
i8* getelementptr inbounds ([18 x i8]* @.str, i32 0, i32 0))
  %weight1 = load i32* %weight
  %printf2 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt4, i32 0, i32
0), i32 %weight1)
  %age3 = load i32* %age
  %printf4 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt8, i32 0, i32
0), i32 %age3)
  br label %pred_msg_while_bb

pred_msg_while_bb:                                ; preds = %entry, %finish_msg_while_bb
  br i1 true, label %body_msg_while_bb, label %merge_msg_while_bb

body_msg_while_bb:                                ; preds = %pred_msg_while_bb
  %"self:index_val" = load i32* %self_index
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%"self:index_val"
  %12 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %13 = load i8** %12
  %14 = bitcast i8* %13 to %struct.head*
  %message_struct = alloca %struct.message
  call void @dequeue(%struct.message* %message_struct, %struct.head* %14)
  %15 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 0
  %case_num = load i32* %15
  %16 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 1
  %actuals_ptr = load i8** %16
  %17 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 2
  %sender_ptr = load i8** %17
  switch i32 %case_num, label %msg_default_case_bb [
    i32 1, label %msg_eat_case_bb
    i32 0, label %msg_die_case_bb
  ]

finish_msg_while_bb:                              ; preds = %msg_eat_case_bb, %msg_default_case_bb
  br label %pred_msg_while_bb
```

```
merge_msg_while_bb:                               ; preds = %msg_die_case_bb, %pred_msg_while_bb
  %ret = alloca i8
  ret i8* %ret

msg_die_case_bb:                                  ; preds = %body_msg_while_bb
  br label %merge_msg_while_bb

msg_eat_case_bb:                                  ; preds = %body_msg_while_bb
  %actual_ptr = bitcast i8* %actuals_ptr to %eat_struct*
  %18 = alloca %eat_struct*
  store %eat_struct* %actual_ptr, %eat_struct** %18
  %19 = load %eat_struct** %18
  %f_ptr = getelementptr inbounds %eat_struct* %19, i32 0, i32 0
  %20 = load i32* %f_ptr
  %food = alloca i32
  store i32 %20, i32* %food
  %weight6 = load i32* %weight
  %food7 = load i32* %food
  %tmp = add i32 %weight6, %food7
  store i32 %tmp, i32* %weight
  %printf8 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt19, i32 0, i32
0), i8* getelementptr inbounds ([32 x i8]* @.str21, i32 0, i32 0))
  %weight9 = load i32* %weight
  %printf10 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt22, i32 0, i32
0), i32 %weight9)
  br label %finish_msg_while_bb

msg_default_case_bb:                              ; preds = %body_msg_while_bb
  %printf5 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt14, i32 0, i32
0), i8* getelementptr inbounds ([14 x i8]* @.str16, i32 0, i32 0))
  br label %finish_msg_while_bb
}

define i8* @dolphin(i8* %ptr) {
entry:
  %0 = alloca i8*
  store i8* %ptr, i8** %0
  %state_struct = alloca %dolphin_struct*
  %1 = load i8** %0
  %2 = bitcast i8* %1 to %dolphin_struct*
  store %dolphin_struct* %2, %dolphin_struct** %state_struct
  %3 = load %dolphin_struct** %state_struct
  %4 = getelementptr inbounds %dolphin_struct* %3, i32 0, i32 0
  %5 = load i32* %4
  %self_index = alloca i32
```

```
  store i32 %5, i32* %self_index
  %6 = load %dolphin_struct** %state_struct
  %7 = getelementptr inbounds %dolphin_struct* %6, i32 0, i32 1
  %8 = load i32* %7
  %weight = alloca i32
  store i32 %8, i32* %weight
  %9 = load %dolphin_struct** %state_struct
  %10 = getelementptr inbounds %dolphin_struct* %9, i32 0, i32 2
  %11 = load i32* %10
  %age = alloca i32
  store i32 %11, i32* %age
  %baby = alloca %actor_address_struct*
  tail call void @free(i8* %ptr)
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt28, i32 0, i32
0), i8* getelementptr inbounds ([13 x i8]* @.str30, i32 0, i32 0))
  %weight1 = load i32* %weight
  %printf2 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt31, i32 0, i32
0), i32 %weight1)
  %age3 = load i32* %age
  %printf4 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt35, i32 0, i32
0), i32 %age3)
  %age5 = load i32* %age
  %tmp = mul i32 %age5, 2
  %weight6 = load i32* %weight
  %tmp7 = mul i32 %weight6, 2
  %curr_actor_count = load i32* @actor_count
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count
  %12 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 0
  store i32 1, i32* %12
  %13 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 1
  %14 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %15 = call %struct.head* @initialize_queue()
  %16 = bitcast %struct.head* %15 to i8*
  store i8* %16, i8** %14
  %malloccall = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i32* getelementptr (i32*
null, i32 1) to i64), i64 3) to i32))
  %17 = bitcast i8* %malloccall to %babyDolphin_struct*
  %18 = getelementptr inbounds %babyDolphin_struct* %17, i32 0, i32 0
  store i32 %curr_actor_count, i32* %18
  %19 = getelementptr inbounds %babyDolphin_struct* %17, i32 0, i32 1
  store i32 %tmp7, i32* %19
  %20 = getelementptr inbounds %babyDolphin_struct* %17, i32 0, i32 2
  store i32 %tmp, i32* %20
  %21 = bitcast %babyDolphin_struct* %17 to i8*
  %tid = alloca i32
```

```
  %pthread_create_result = call i32 @pthread_create(i32* %tid, i8* null, i8* (i8*)* @babyDolphin,
i8* %21)
  %22 = load i32* %tid
  store i32 %22, i32* %13
  %23 = add i32 %curr_actor_count, 1
  store i32 %23, i32* @actor_count
  store %actor_address_struct* %pos, %actor_address_struct** %baby
  %addr_struct = load %actor_address_struct** %baby
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %24 = bitcast i8* %msgQueue to %struct.head*
  %25 = alloca %struct.message
  %malloccall8 = tail call i8* @malloc(i32 0)
  %argument_struct = bitcast i8* %malloccall8 to %temp*
  %case = alloca i32
  store i32 0, i32* %case
  %26 = getelementptr inbounds %struct.message* %25, i32 0, i32 0
  %27 = getelementptr inbounds %struct.message* %25, i32 0, i32 1
  %28 = getelementptr inbounds %struct.message* %25, i32 0, i32 2
  store i32 0, i32* %26
  store i8* null, i8** %27
  store i8* null, i8** %28
  %message_val = load %struct.message* %25
  call void @enqueue(%struct.head* %24, %struct.message %message_val)
  br label %pred_msg_while_bb

pred_msg_while_bb:                                ; preds = %entry, %finish_msg_while_bb
  br i1 true, label %body_msg_while_bb, label %merge_msg_while_bb

body_msg_while_bb:                                ; preds = %pred_msg_while_bb
  %"self:index_val" = load i32* %self_index
  %pos9 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%"self:index_val"
  %29 = getelementptr inbounds %actor_address_struct* %pos9, i32 0, i32 2
  %30 = load i8** %29
  %31 = bitcast i8* %30 to %struct.head*
  %message_struct = alloca %struct.message
  call void @dequeue(%struct.message* %message_struct, %struct.head* %31)
  %32 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 0
  %case_num = load i32* %32
  %33 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 1
  %actuals_ptr = load i8** %33
  %34 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 2
  %sender_ptr = load i8** %34
  switch i32 %case_num, label %msg_default_case_bb [
    i32 0, label %msg_die_case_bb
```

```
    i32 2, label %msg_eat_case_bb
    i32 4, label %msg_eatAndGrowOld_case_bb
    i32 3, label %msg_growOld_case_bb
  ]

finish_msg_while_bb:                              ; preds = %msg_eatAndGrowOld_case_bb,
%msg_growOld_case_bb, %msg_eat_case_bb, %msg_default_case_bb
  br label %pred_msg_while_bb

merge_msg_while_bb:                               ; preds = %msg_die_case_bb, %pred_msg_while_bb
  %ret = alloca i8
  ret i8* %ret

msg_die_case_bb:                                  ; preds = %body_msg_while_bb
  br label %merge_msg_while_bb

msg_eat_case_bb:                                  ; preds = %body_msg_while_bb
  %actual_ptr = bitcast i8* %actuals_ptr to %eat_struct.0*
  %35 = alloca %eat_struct.0*
  store %eat_struct.0* %actual_ptr, %eat_struct.0** %35
  %36 = load %eat_struct.0** %35
  %f_ptr = getelementptr inbounds %eat_struct.0* %36, i32 0, i32 0
  %37 = load i32* %f_ptr
  %food = alloca i32
  store i32 %37, i32* %food
  %weight11 = load i32* %weight
  %food12 = load i32* %food
  %tmp13 = add i32 %weight11, %food12
  store i32 %tmp13, i32* %weight
  %printf14 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt46, i32 0, i32
0), i8* getelementptr inbounds ([27 x i8]* @.str48, i32 0, i32 0))
  %weight15 = load i32* %weight
  %printf16 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt49, i32 0, i32
0), i32 %weight15)
  br label %finish_msg_while_bb

msg_growOld_case_bb:                              ; preds = %body_msg_while_bb
  %actual_ptr17 = bitcast i8* %actuals_ptr to %growOld_struct*
  %38 = alloca %growOld_struct*
  store %growOld_struct* %actual_ptr17, %growOld_struct** %38
  %39 = load %growOld_struct** %38
  %f_ptr18 = getelementptr inbounds %growOld_struct* %39, i32 0, i32 0
  %40 = load i32* %f_ptr18
  %time = alloca i32
  store i32 %40, i32* %time
  %age19 = load i32* %age
```

```
  %time20 = load i32* %time
  %tmp21 = add i32 %age19, %time20
  store i32 %tmp21, i32* %age
  br label %finish_msg_while_bb

msg_eatAndGrowOld_case_bb:                        ; preds = %body_msg_while_bb
  %actual_ptr22 = bitcast i8* %actuals_ptr to %eatAndGrowOld_struct*
  %41 = alloca %eatAndGrowOld_struct*
  store %eatAndGrowOld_struct* %actual_ptr22, %eatAndGrowOld_struct** %41
  %42 = load %eatAndGrowOld_struct** %41
  %f_ptr23 = getelementptr inbounds %eatAndGrowOld_struct* %42, i32 0, i32 0
  %43 = load i32* %f_ptr23
  %food24 = alloca i32
  store i32 %43, i32* %food24
  %f_ptr25 = getelementptr inbounds %eatAndGrowOld_struct* %42, i32 0, i32 1
  %44 = load i32* %f_ptr25
  %time26 = alloca i32
  store i32 %44, i32* %time26
  %weight27 = load i32* %weight
  %food28 = load i32* %food24
  %tmp29 = add i32 %weight27, %food28
  store i32 %tmp29, i32* %weight
  %age30 = load i32* %age
  %time31 = load i32* %time26
  %tmp32 = add i32 %age30, %time31
  store i32 %tmp32, i32* %age
  br label %finish_msg_while_bb

msg_default_case_bb:                              ; preds = %body_msg_while_bb
  %printf10 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt41, i32 0, i32
0), i8* getelementptr inbounds ([14 x i8]* @.str43, i32 0, i32 0))
  br label %finish_msg_while_bb
}

declare void @free(i8*)

declare noalias i8* @malloc(i32)
```

Switchboard simulation

This example simulates a switchboard operator whose job it is to connect calls between callers. In order to place a call, a caller notifies the operator by passing it a message, including a handle

to the callee. The operator facilitates the call by passing messages between the caller and callee to set up the connection. This demonstrates the feature of passing along handles to actors via messages and showcases what you could do with multiple concurrent actors being synchronized by a controller actor.

```
int totalHangups

operator(int capacity):
    int count = 0
    receive:
        tryConnect(actor src, int srcId, actor dest, int destId):
            print("OPERATOR: trying to make a connection")
            if (count < capacity):
                print("OPERATOR: have capacity, checking if callee is busy")
                caller.checkIfBusy(dest, src, srcId) | dest
                count = count + 1
            else:
                print("OPERATOR: at capacity, denying")
                caller.cantConnect(src, dest) | src
        respondBusy(actor dest, int destId, actor src, int srcId):
            print("OPERATOR: allee is busy")
            caller.cantConnect(src, dest) | src
            count = count - 1
        respondOK(actor dest, int destId, actor src, int srcId):
            print("OPERATOR: making a connection")
            caller.connect(src, dest, destId) | src
        endCall(actor src, int srcId, actor dest, int destId):
            print("OPERATOR: disconnecting a call")
            caller.disconnect() | dest
            count = count + 1
    drop:
        print("invalid function, ignoring")

caller(actor op, int id):
    actor partner
    int partnerId = 9999
    int busy = 0
    receive:
        makeCall(actor me, actor dest, int destId):
            print("CALLER: trying to make call")
            if (busy == 0):
                print("CALLER: tryConnect to operator")
                busy = 1
                partnerId = destId
                partner = dest
```

```
                operator.tryConnect(me, id, dest, partnerId) | op
        checkIfBusy(actor me, actor caller, int callerId):
            print("CALLER: checking if busy")
            if (busy == 1):
                print("CALLER: busy, respondBusy to operator")
                operator.respondBusy(me, id, caller, callerId) | op
            else:
                print("CALLER: free, respondOK to operator")
                operator.respondOK(me, id, caller, callerId) | op
                busy = 1
                partnerId = callerId
                partner = caller
        cantConnect(actor me, actor dest):
            print("CALLER: not busy anymore")
            busy = 0
            partnerId = 9999
        connect(actor me, actor dest, int destId):
            print("CALLER: Connected and talking now")
            busy = 1
            partnerId = destId
            partner = dest
        hangUp(actor me):
            print("CALLER: Got hangup command")
            if (busy == 1):
                print("CALLER: Hanging up on partner")
                busy = 0
                operator.endCall(me, id, partner, partnerId) | op
                partnerId = 9999
        disconnect():
            print("CALLER: Got disconnected from")
            busy = 0
            partnerId = 9999
            totalHangups = totalHangups + 1
    drop:
        print("invalid function, ignoring")

func main() -> int:
    totalHangups = 0
    print("operator capacity: ")
    print(3)
    actor o = new operator(3)
    actor c1 = new caller(o, 1)
    actor c2 = new caller(o, 2)
    actor c3 = new caller(o, 3)
    actor c4 = new caller(o, 4)
    actor c5 = new caller(o, 5)
```

```
    caller.makeCall(c1, c2, 2) | c1
    caller.makeCall(c3, c4, 4) | c3
    caller.makeCall(c5, c1, 1) | c5

    int i = 0
    while(i < 4000000):
        i = i + 1
    caller.hangUp(c4) | c4
    caller.hangUp(c2) | c2
    int j = 0
    while(totalHangups < 2):
        j = 0
    caller.die() | c1
    caller.die() | c2
    caller.die() | c3
    caller.die() | c4
    caller.die() | c5
    operator.die() | o

    return 0
```

The resulting LLVM code is:

```
; ModuleID = 'Theatr'

%actor_address_struct = type { i32, i32, i8* }
%struct.head = type { %struct.queue*, %union.pthread_cond_t, %union.pthread_mutex_t }
%struct.queue = type { %struct.queue*, %struct.message }
%struct.message = type { i32, i8*, i8* }
%union.pthread_cond_t = type { %struct.anon, [4 x i8*] }
%struct.anon = type { i32, i32, i64, i64, i64, i8*, i32, i32 }
%union.pthread_mutex_t = type { %struct.__pthread_mutex_s }
%struct.__pthread_mutex_s = type { i32, i32, i32, i32, i32, %union.anon }
%union.anon = type { %struct.__pthread_internal_slist }
%struct.__pthread_internal_slist = type { %struct.__pthread_internal_slist* }
%operator_struct = type { i32, i32 }
%caller_struct = type { i32, %actor_address_struct*, i32 }
%temp.8 = type { %actor_address_struct*, %actor_address_struct*, i32 }
%temp.9 = type { %actor_address_struct*, %actor_address_struct*, i32 }
%temp.10 = type { %actor_address_struct*, %actor_address_struct*, i32 }
%temp.11 = type { %actor_address_struct* }
%temp.12 = type { %actor_address_struct* }
%temp.13 = type {}
%temp.14 = type {}
%temp.15 = type {}
%temp.16 = type {}
```

```
%temp.17 = type {}
%temp.18 = type {}
%makeCall_struct = type { %actor_address_struct*, %actor_address_struct*, i32 }
%checkIfBusy_struct = type { %actor_address_struct*, %actor_address_struct*, i32 }
%cantConnect_struct = type { %actor_address_struct*, %actor_address_struct* }
%connect_struct = type { %actor_address_struct*, %actor_address_struct*, i32 }
%hangUp_struct = type { %actor_address_struct* }
%disconnect_struct = type {}
%temp = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.0 = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.1 = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.2 = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%tryConnect_struct = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%respondBusy_struct = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.5 = type { %actor_address_struct*, %actor_address_struct* }
%respondOK_struct = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.6 = type { %actor_address_struct*, %actor_address_struct*, i32 }
%endCall_struct = type { %actor_address_struct*, i32, %actor_address_struct*, i32 }
%temp.7 = type {}
%temp.3 = type { %actor_address_struct*, %actor_address_struct*, i32 }
%temp.4 = type { %actor_address_struct*, %actor_address_struct* }

@global_actors = global [1024 x %actor_address_struct] zeroinitializer
@actor_count = global i32 1
@totalHangups = global i32 0
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt3 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str = private unnamed_addr constant [27 x i8] c"invalid function, ignoring\00"
@fmt4 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt5 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt6 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt7 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str8 = private unnamed_addr constant [28 x i8] c"CALLER: trying to make call\00"
@fmt9 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt10 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt11 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt12 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str13 = private unnamed_addr constant [31 x i8] c"CALLER: tryConnect to operator\00"
@fmt14 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt15 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt16 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt17 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str18 = private unnamed_addr constant [25 x i8] c"CALLER: checking if busy\00"
@fmt19 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
```

```
@fmt20 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt21 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt22 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str23 = private unnamed_addr constant [38 x i8] c"CALLER: busy, respondBusy to operator\00"
@fmt24 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt25 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt26 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt27 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str28 = private unnamed_addr constant [36 x i8] c"CALLER: free, respondOK to operator\00"
@fmt29 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt30 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt31 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt32 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str33 = private unnamed_addr constant [25 x i8] c"CALLER: not busy anymore\00"
@fmt34 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt35 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt36 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt37 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str38 = private unnamed_addr constant [34 x i8] c"CALLER: Connected and talking now\00"
@fmt39 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt40 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt41 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt42 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str43 = private unnamed_addr constant [27 x i8] c"CALLER: Got hangup command\00"
@fmt44 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt45 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt46 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt47 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str48 = private unnamed_addr constant [30 x i8] c"CALLER: Hanging up on partner\00"
@fmt49 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt50 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt51 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt52 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str53 = private unnamed_addr constant [30 x i8] c"CALLER: Got disconnected from\00"
@fmt54 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt55 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt56 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt57 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str58 = private unnamed_addr constant [27 x i8] c"invalid function, ignoring\00"
@fmt59 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt60 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt61 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt62 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str63 = private unnamed_addr constant [38 x i8] c"OPERATOR: trying to make a connection\00"
@fmt64 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt65 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
```

```
@fmt66 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt67 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str68 = private unnamed_addr constant [52 x i8] c"OPERATOR: have capacity, checking if callee is
busy\00"
@fmt69 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt70 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt71 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt72 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str73 = private unnamed_addr constant [31 x i8] c"OPERATOR: at capacity, denying\00"
@fmt74 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt75 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt76 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt77 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str78 = private unnamed_addr constant [24 x i8] c"OPERATOR: allee is busy\00"
@fmt79 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt80 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt81 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt82 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str83 = private unnamed_addr constant [30 x i8] c"OPERATOR: making a connection\00"
@fmt84 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt85 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt86 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt87 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str88 = private unnamed_addr constant [31 x i8] c"OPERATOR: disconnecting a call\00"
@fmt89 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt90 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt91 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt92 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
@.str93 = private unnamed_addr constant [20 x i8] c"operator capacity: \00"
@fmt94 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt95 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@fmt96 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt97 = private unnamed_addr constant [4 x i8] c"%c\0A\00"

declare %struct.head* @initialize_queue()

declare void @enqueue(%struct.head*, %struct.message)

declare void @dequeue(%struct.message*, %struct.head*)

declare i32 @geturl(i8*, i8*)

declare i32 @printf(i8*, ...)

declare i32 @pthread_create(i32*, i8*, i8* (i8*)*, i8*)
```

```
declare i32 @pthread_join(i32, i8**)

define i32 @main() {
entry:
  %o = alloca %actor_address_struct*
  %c1 = alloca %actor_address_struct*
  %c2 = alloca %actor_address_struct*
  %c3 = alloca %actor_address_struct*
  %c4 = alloca %actor_address_struct*
  %c5 = alloca %actor_address_struct*
  %i = alloca i32
  %j = alloca i32
  store i32 0, i32* @totalHangups
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt91, i32 0, i32
0), i8* getelementptr inbounds ([20 x i8]* @.str93, i32 0, i32 0))
  %printf1 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt94, i32 0, i32
0), i32 3)
  %curr_actor_count = load i32* @actor_count
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count
  %0 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 0
  store i32 1, i32* %0
  %1 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 1
  %2 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %3 = call %struct.head* @initialize_queue()
  %4 = bitcast %struct.head* %3 to i8*
  store i8* %4, i8** %2
  %malloccall = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i32* getelementptr (i32*
null, i32 1) to i64), i64 2) to i32))
  %5 = bitcast i8* %malloccall to %operator_struct*
  %6 = getelementptr inbounds %operator_struct* %5, i32 0, i32 0
  store i32 %curr_actor_count, i32* %6
  %7 = getelementptr inbounds %operator_struct* %5, i32 0, i32 1
  store i32 3, i32* %7
  %8 = bitcast %operator_struct* %5 to i8*
  %tid = alloca i32
  %pthread_create_result = call i32 @pthread_create(i32* %tid, i8* null, i8* (i8*)* @operator, i8*
%8)
  %9 = load i32* %tid
  store i32 %9, i32* %1
  %10 = add i32 %curr_actor_count, 1
  store i32 %10, i32* @actor_count
  store %actor_address_struct* %pos, %actor_address_struct** %o
  %o2 = load %actor_address_struct** %o
  %curr_actor_count3 = load i32* @actor_count
  %pos4 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
```

```
%curr_actor_count3
  %11 = getelementptr inbounds %actor_address_struct* %pos4, i32 0, i32 0
  store i32 1, i32* %11
  %12 = getelementptr inbounds %actor_address_struct* %pos4, i32 0, i32 1
  %13 = getelementptr inbounds %actor_address_struct* %pos4, i32 0, i32 2
  %14 = call %struct.head* @initialize_queue()
  %15 = bitcast %struct.head* %14 to i8*
  store i8* %15, i8** %13
  %malloccall5 = tail call i8* @malloc(i32 ptrtoint (%caller_struct* getelementptr (%caller_struct*
null, i32 1) to i32))
  %16 = bitcast i8* %malloccall5 to %caller_struct*
  %17 = getelementptr inbounds %caller_struct* %16, i32 0, i32 0
  store i32 %curr_actor_count3, i32* %17
  %18 = getelementptr inbounds %caller_struct* %16, i32 0, i32 1
  store %actor_address_struct* %o2, %actor_address_struct** %18
  %19 = getelementptr inbounds %caller_struct* %16, i32 0, i32 2
  store i32 1, i32* %19
  %20 = bitcast %caller_struct* %16 to i8*
  %tid6 = alloca i32
  %pthread_create_result7 = call i32 @pthread_create(i32* %tid6, i8* null, i8* (i8*)* @caller, i8*
%20)
  %21 = load i32* %tid6
  store i32 %21, i32* %12
  %22 = add i32 %curr_actor_count3, 1
  store i32 %22, i32* @actor_count
  store %actor_address_struct* %pos4, %actor_address_struct** %c1
  %o8 = load %actor_address_struct** %o
  %curr_actor_count9 = load i32* @actor_count
  %pos10 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count9
  %23 = getelementptr inbounds %actor_address_struct* %pos10, i32 0, i32 0
  store i32 1, i32* %23
  %24 = getelementptr inbounds %actor_address_struct* %pos10, i32 0, i32 1
  %25 = getelementptr inbounds %actor_address_struct* %pos10, i32 0, i32 2
  %26 = call %struct.head* @initialize_queue()
  %27 = bitcast %struct.head* %26 to i8*
  store i8* %27, i8** %25
  %malloccall11 = tail call i8* @malloc(i32 ptrtoint (%caller_struct* getelementptr (%caller_struct*
null, i32 1) to i32))
  %28 = bitcast i8* %malloccall11 to %caller_struct*
  %29 = getelementptr inbounds %caller_struct* %28, i32 0, i32 0
  store i32 %curr_actor_count9, i32* %29
  %30 = getelementptr inbounds %caller_struct* %28, i32 0, i32 1
  store %actor_address_struct* %o8, %actor_address_struct** %30
  %31 = getelementptr inbounds %caller_struct* %28, i32 0, i32 2
  store i32 2, i32* %31
```

```
  %32 = bitcast %caller_struct* %28 to i8*
  %tid12 = alloca i32
  %pthread_create_result13 = call i32 @pthread_create(i32* %tid12, i8* null, i8* (i8*)* @caller, i8*
%32)
  %33 = load i32* %tid12
  store i32 %33, i32* %24
  %34 = add i32 %curr_actor_count9, 1
  store i32 %34, i32* @actor_count
  store %actor_address_struct* %pos10, %actor_address_struct** %c2
  %o14 = load %actor_address_struct** %o
  %curr_actor_count15 = load i32* @actor_count
  %pos16 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count15
  %35 = getelementptr inbounds %actor_address_struct* %pos16, i32 0, i32 0
  store i32 1, i32* %35
  %36 = getelementptr inbounds %actor_address_struct* %pos16, i32 0, i32 1
  %37 = getelementptr inbounds %actor_address_struct* %pos16, i32 0, i32 2
  %38 = call %struct.head* @initialize_queue()
  %39 = bitcast %struct.head* %38 to i8*
  store i8* %39, i8** %37
  %malloccall17 = tail call i8* @malloc(i32 ptrtoint (%caller_struct* getelementptr (%caller_struct*
null, i32 1) to i32))
  %40 = bitcast i8* %malloccall17 to %caller_struct*
  %41 = getelementptr inbounds %caller_struct* %40, i32 0, i32 0
  store i32 %curr_actor_count15, i32* %41
  %42 = getelementptr inbounds %caller_struct* %40, i32 0, i32 1
  store %actor_address_struct* %o14, %actor_address_struct** %42
  %43 = getelementptr inbounds %caller_struct* %40, i32 0, i32 2
  store i32 3, i32* %43
  %44 = bitcast %caller_struct* %40 to i8*
  %tid18 = alloca i32
  %pthread_create_result19 = call i32 @pthread_create(i32* %tid18, i8* null, i8* (i8*)* @caller, i8*
%44)
  %45 = load i32* %tid18
  store i32 %45, i32* %36
  %46 = add i32 %curr_actor_count15, 1
  store i32 %46, i32* @actor_count
  store %actor_address_struct* %pos16, %actor_address_struct** %c3
  %o20 = load %actor_address_struct** %o
  %curr_actor_count21 = load i32* @actor_count
  %pos22 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count21
  %47 = getelementptr inbounds %actor_address_struct* %pos22, i32 0, i32 0
  store i32 1, i32* %47
  %48 = getelementptr inbounds %actor_address_struct* %pos22, i32 0, i32 1
  %49 = getelementptr inbounds %actor_address_struct* %pos22, i32 0, i32 2
```

```
  %50 = call %struct.head* @initialize_queue()
  %51 = bitcast %struct.head* %50 to i8*
  store i8* %51, i8** %49
  %malloccall23 = tail call i8* @malloc(i32 ptrtoint (%caller_struct* getelementptr (%caller_struct*
null, i32 1) to i32))
  %52 = bitcast i8* %malloccall23 to %caller_struct*
  %53 = getelementptr inbounds %caller_struct* %52, i32 0, i32 0
  store i32 %curr_actor_count21, i32* %53
  %54 = getelementptr inbounds %caller_struct* %52, i32 0, i32 1
  store %actor_address_struct* %o20, %actor_address_struct** %54
  %55 = getelementptr inbounds %caller_struct* %52, i32 0, i32 2
  store i32 4, i32* %55
  %56 = bitcast %caller_struct* %52 to i8*
  %tid24 = alloca i32
  %pthread_create_result25 = call i32 @pthread_create(i32* %tid24, i8* null, i8* (i8*)* @caller, i8*
%56)
  %57 = load i32* %tid24
  store i32 %57, i32* %48
  %58 = add i32 %curr_actor_count21, 1
  store i32 %58, i32* @actor_count
  store %actor_address_struct* %pos22, %actor_address_struct** %c4
  %o26 = load %actor_address_struct** %o
  %curr_actor_count27 = load i32* @actor_count
  %pos28 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%curr_actor_count27
  %59 = getelementptr inbounds %actor_address_struct* %pos28, i32 0, i32 0
  store i32 1, i32* %59
  %60 = getelementptr inbounds %actor_address_struct* %pos28, i32 0, i32 1
  %61 = getelementptr inbounds %actor_address_struct* %pos28, i32 0, i32 2
  %62 = call %struct.head* @initialize_queue()
  %63 = bitcast %struct.head* %62 to i8*
  store i8* %63, i8** %61
  %malloccall29 = tail call i8* @malloc(i32 ptrtoint (%caller_struct* getelementptr (%caller_struct*
null, i32 1) to i32))
  %64 = bitcast i8* %malloccall29 to %caller_struct*
  %65 = getelementptr inbounds %caller_struct* %64, i32 0, i32 0
  store i32 %curr_actor_count27, i32* %65
  %66 = getelementptr inbounds %caller_struct* %64, i32 0, i32 1
  store %actor_address_struct* %o26, %actor_address_struct** %66
  %67 = getelementptr inbounds %caller_struct* %64, i32 0, i32 2
  store i32 5, i32* %67
  %68 = bitcast %caller_struct* %64 to i8*
  %tid30 = alloca i32
  %pthread_create_result31 = call i32 @pthread_create(i32* %tid30, i8* null, i8* (i8*)* @caller, i8*
%68)
  %69 = load i32* %tid30
```

```
  store i32 %69, i32* %60
  %70 = add i32 %curr_actor_count27, 1
  store i32 %70, i32* @actor_count
  store %actor_address_struct* %pos28, %actor_address_struct** %c5
  %addr_struct = load %actor_address_struct** %c1
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %71 = bitcast i8* %msgQueue to %struct.head*
  %72 = alloca %struct.message
  %c232 = load %actor_address_struct** %c2
  %c133 = load %actor_address_struct** %c1
  %malloccall34 = tail call i8* @malloc(i32 ptrtoint (%temp.8* getelementptr (%temp.8* null, i32 1)
to i32))
  %argument_struct = bitcast i8* %malloccall34 to %temp.8*
  %73 = alloca %actor_address_struct*
  store %actor_address_struct* %c133, %actor_address_struct** %73
  %arg_val = load %actor_address_struct** %73
  %74 = getelementptr inbounds %temp.8* %argument_struct, i32 0, i32 0
  store %actor_address_struct* %arg_val, %actor_address_struct** %74
  %75 = alloca %actor_address_struct*
  store %actor_address_struct* %c232, %actor_address_struct** %75
  %arg_val35 = load %actor_address_struct** %75
  %76 = getelementptr inbounds %temp.8* %argument_struct, i32 0, i32 1
  store %actor_address_struct* %arg_val35, %actor_address_struct** %76
  %77 = alloca i32
  store i32 2, i32* %77
  %arg_val36 = load i32* %77
  %78 = getelementptr inbounds %temp.8* %argument_struct, i32 0, i32 2
  store i32 %arg_val36, i32* %78
  %79 = bitcast %temp.8* %argument_struct to i8*
  %case = alloca i32
  store i32 1, i32* %case
  %80 = getelementptr inbounds %struct.message* %72, i32 0, i32 0
  %81 = getelementptr inbounds %struct.message* %72, i32 0, i32 1
  %82 = getelementptr inbounds %struct.message* %72, i32 0, i32 2
  store i32 1, i32* %80
  store i8* %79, i8** %81
  store i8* null, i8** %82
  %message_val = load %struct.message* %72
  call void @enqueue(%struct.head* %71, %struct.message %message_val)
  %addr_struct37 = load %actor_address_struct** %c3
  %msgQueue_ptr38 = getelementptr inbounds %actor_address_struct* %addr_struct37, i32 0, i32 2
  %msgQueue39 = load i8** %msgQueue_ptr38
  %83 = bitcast i8* %msgQueue39 to %struct.head*
  %84 = alloca %struct.message
  %c440 = load %actor_address_struct** %c4
```

```
  %c341 = load %actor_address_struct** %c3
  %malloccall42 = tail call i8* @malloc(i32 ptrtoint (%temp.9* getelementptr (%temp.9* null, i32 1)
to i32))
  %argument_struct43 = bitcast i8* %malloccall42 to %temp.9*
  %85 = alloca %actor_address_struct*
  store %actor_address_struct* %c341, %actor_address_struct** %85
  %arg_val44 = load %actor_address_struct** %85
  %86 = getelementptr inbounds %temp.9* %argument_struct43, i32 0, i32 0
  store %actor_address_struct* %arg_val44, %actor_address_struct** %86
  %87 = alloca %actor_address_struct*
  store %actor_address_struct* %c440, %actor_address_struct** %87
  %arg_val45 = load %actor_address_struct** %87
  %88 = getelementptr inbounds %temp.9* %argument_struct43, i32 0, i32 1
  store %actor_address_struct* %arg_val45, %actor_address_struct** %88
  %89 = alloca i32
  store i32 4, i32* %89
  %arg_val46 = load i32* %89
  %90 = getelementptr inbounds %temp.9* %argument_struct43, i32 0, i32 2
  store i32 %arg_val46, i32* %90
  %91 = bitcast %temp.9* %argument_struct43 to i8*
  %case47 = alloca i32
  store i32 1, i32* %case47
  %92 = getelementptr inbounds %struct.message* %84, i32 0, i32 0
  %93 = getelementptr inbounds %struct.message* %84, i32 0, i32 1
  %94 = getelementptr inbounds %struct.message* %84, i32 0, i32 2
  store i32 1, i32* %92
  store i8* %91, i8** %93
  store i8* null, i8** %94
  %message_val48 = load %struct.message* %84
  call void @enqueue(%struct.head* %83, %struct.message %message_val48)
  %addr_struct49 = load %actor_address_struct** %c5
  %msgQueue_ptr50 = getelementptr inbounds %actor_address_struct* %addr_struct49, i32 0, i32 2
  %msgQueue51 = load i8** %msgQueue_ptr50
  %95 = bitcast i8* %msgQueue51 to %struct.head*
  %96 = alloca %struct.message
  %c152 = load %actor_address_struct** %c1
  %c553 = load %actor_address_struct** %c5
  %malloccall54 = tail call i8* @malloc(i32 ptrtoint (%temp.10* getelementptr (%temp.10* null, i32
1) to i32))
  %argument_struct55 = bitcast i8* %malloccall54 to %temp.10*
  %97 = alloca %actor_address_struct*
  store %actor_address_struct* %c553, %actor_address_struct** %97
  %arg_val56 = load %actor_address_struct** %97
  %98 = getelementptr inbounds %temp.10* %argument_struct55, i32 0, i32 0
  store %actor_address_struct* %arg_val56, %actor_address_struct** %98
  %99 = alloca %actor_address_struct*
```

```
  store %actor_address_struct* %c152, %actor_address_struct** %99
  %arg_val57 = load %actor_address_struct** %99
  %100 = getelementptr inbounds %temp.10* %argument_struct55, i32 0, i32 1
  store %actor_address_struct* %arg_val57, %actor_address_struct** %100
  %101 = alloca i32
  store i32 1, i32* %101
  %arg_val58 = load i32* %101
  %102 = getelementptr inbounds %temp.10* %argument_struct55, i32 0, i32 2
  store i32 %arg_val58, i32* %102
  %103 = bitcast %temp.10* %argument_struct55 to i8*
  %case59 = alloca i32
  store i32 1, i32* %case59
  %104 = getelementptr inbounds %struct.message* %96, i32 0, i32 0
  %105 = getelementptr inbounds %struct.message* %96, i32 0, i32 1
  %106 = getelementptr inbounds %struct.message* %96, i32 0, i32 2
  store i32 1, i32* %104
  store i8* %103, i8** %105
  store i8* null, i8** %106
  %message_val60 = load %struct.message* %96
  call void @enqueue(%struct.head* %95, %struct.message %message_val60)
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %while_body, %entry
  %i62 = load i32* %i
  %tmp63 = icmp slt i32 %i62, 4000000
  br i1 %tmp63, label %while_body, label %merge

while_body:                                       ; preds = %while
  %i61 = load i32* %i
  %tmp = add i32 %i61, 1
  store i32 %tmp, i32* %i
  br label %while

merge:                                            ; preds = %while
  %addr_struct64 = load %actor_address_struct** %c4
  %msgQueue_ptr65 = getelementptr inbounds %actor_address_struct* %addr_struct64, i32 0, i32 2
  %msgQueue66 = load i8** %msgQueue_ptr65
  %107 = bitcast i8* %msgQueue66 to %struct.head*
  %108 = alloca %struct.message
  %c467 = load %actor_address_struct** %c4
  %malloccall68 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to i32))
  %argument_struct69 = bitcast i8* %malloccall68 to %temp.11*
  %109 = alloca %actor_address_struct*
  store %actor_address_struct* %c467, %actor_address_struct** %109
  %arg_val70 = load %actor_address_struct** %109
```

```
  %110 = getelementptr inbounds %temp.11* %argument_struct69, i32 0, i32 0
  store %actor_address_struct* %arg_val70, %actor_address_struct** %110
  %111 = bitcast %temp.11* %argument_struct69 to i8*
  %case71 = alloca i32
  store i32 5, i32* %case71
  %112 = getelementptr inbounds %struct.message* %108, i32 0, i32 0
  %113 = getelementptr inbounds %struct.message* %108, i32 0, i32 1
  %114 = getelementptr inbounds %struct.message* %108, i32 0, i32 2
  store i32 5, i32* %112
  store i8* %111, i8** %113
  store i8* null, i8** %114
  %message_val72 = load %struct.message* %108
  call void @enqueue(%struct.head* %107, %struct.message %message_val72)
  %addr_struct73 = load %actor_address_struct** %c2
  %msgQueue_ptr74 = getelementptr inbounds %actor_address_struct* %addr_struct73, i32 0, i32 2
  %msgQueue75 = load i8** %msgQueue_ptr74
  %115 = bitcast i8* %msgQueue75 to %struct.head*
  %116 = alloca %struct.message
  %c276 = load %actor_address_struct** %c2
  %malloccall77 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1** null, i32 1) to i32))
  %argument_struct78 = bitcast i8* %malloccall77 to %temp.12*
  %117 = alloca %actor_address_struct*
  store %actor_address_struct* %c276, %actor_address_struct** %117
  %arg_val79 = load %actor_address_struct** %117
  %118 = getelementptr inbounds %temp.12* %argument_struct78, i32 0, i32 0
  store %actor_address_struct* %arg_val79, %actor_address_struct** %118
  %119 = bitcast %temp.12* %argument_struct78 to i8*
  %case80 = alloca i32
  store i32 5, i32* %case80
  %120 = getelementptr inbounds %struct.message* %116, i32 0, i32 0
  %121 = getelementptr inbounds %struct.message* %116, i32 0, i32 1
  %122 = getelementptr inbounds %struct.message* %116, i32 0, i32 2
  store i32 5, i32* %120
  store i8* %119, i8** %121
  store i8* null, i8** %122
  %message_val81 = load %struct.message* %116
  call void @enqueue(%struct.head* %115, %struct.message %message_val81)
  store i32 0, i32* %j
  br label %while82

while82:                                          ; preds = %while_body83, %merge
  %totalHangups = load i32* @totalHangups
  %tmp84 = icmp slt i32 %totalHangups, 2
  br i1 %tmp84, label %while_body83, label %merge85

while_body83:                                     ; preds = %while82
```

```
  store i32 0, i32* %j
  br label %while82

merge85:                                          ; preds = %while82
  %addr_struct86 = load %actor_address_struct** %c1
  %msgQueue_ptr87 = getelementptr inbounds %actor_address_struct* %addr_struct86, i32 0, i32 2
  %msgQueue88 = load i8** %msgQueue_ptr87
  %123 = bitcast i8* %msgQueue88 to %struct.head*
  %124 = alloca %struct.message
  %malloccall89 = tail call i8* @malloc(i32 0)
  %argument_struct90 = bitcast i8* %malloccall89 to %temp.13*
  %case91 = alloca i32
  store i32 0, i32* %case91
  %125 = getelementptr inbounds %struct.message* %124, i32 0, i32 0
  %126 = getelementptr inbounds %struct.message* %124, i32 0, i32 1
  %127 = getelementptr inbounds %struct.message* %124, i32 0, i32 2
  store i32 0, i32* %125
  store i8* null, i8** %126
  store i8* null, i8** %127
  %message_val92 = load %struct.message* %124
  call void @enqueue(%struct.head* %123, %struct.message %message_val92)
  %addr_struct93 = load %actor_address_struct** %c2
  %msgQueue_ptr94 = getelementptr inbounds %actor_address_struct* %addr_struct93, i32 0, i32 2
  %msgQueue95 = load i8** %msgQueue_ptr94
  %128 = bitcast i8* %msgQueue95 to %struct.head*
  %129 = alloca %struct.message
  %malloccall96 = tail call i8* @malloc(i32 0)
  %argument_struct97 = bitcast i8* %malloccall96 to %temp.14*
  %case98 = alloca i32
  store i32 0, i32* %case98
  %130 = getelementptr inbounds %struct.message* %129, i32 0, i32 0
  %131 = getelementptr inbounds %struct.message* %129, i32 0, i32 1
  %132 = getelementptr inbounds %struct.message* %129, i32 0, i32 2
  store i32 0, i32* %130
  store i8* null, i8** %131
  store i8* null, i8** %132
  %message_val99 = load %struct.message* %129
  call void @enqueue(%struct.head* %128, %struct.message %message_val99)
  %addr_struct100 = load %actor_address_struct** %c3
  %msgQueue_ptr101 = getelementptr inbounds %actor_address_struct* %addr_struct100, i32 0, i32 2
  %msgQueue102 = load i8** %msgQueue_ptr101
  %133 = bitcast i8* %msgQueue102 to %struct.head*
  %134 = alloca %struct.message
  %malloccall103 = tail call i8* @malloc(i32 0)
  %argument_struct104 = bitcast i8* %malloccall103 to %temp.15*
  %case105 = alloca i32
```

```
store i32 0, i32* %case105
%135 = getelementptr inbounds %struct.message* %134, i32 0, i32 0
%136 = getelementptr inbounds %struct.message* %134, i32 0, i32 1
%137 = getelementptr inbounds %struct.message* %134, i32 0, i32 2
store i32 0, i32* %135
store i8* null, i8** %136
store i8* null, i8** %137
%message_val106 = load %struct.message* %134
call void @enqueue(%struct.head* %133, %struct.message %message_val106)
%addr_struct107 = load %actor_address_struct** %c4
%msgQueue_ptr108 = getelementptr inbounds %actor_address_struct* %addr_struct107, i32 0, i32 2
%msgQueue109 = load i8** %msgQueue_ptr108
%138 = bitcast i8* %msgQueue109 to %struct.head*
%139 = alloca %struct.message
%malloccall110 = tail call i8* @malloc(i32 0)
%argument_struct111 = bitcast i8* %malloccall110 to %temp.16*
%case112 = alloca i32
store i32 0, i32* %case112
%140 = getelementptr inbounds %struct.message* %139, i32 0, i32 0
%141 = getelementptr inbounds %struct.message* %139, i32 0, i32 1
%142 = getelementptr inbounds %struct.message* %139, i32 0, i32 2
store i32 0, i32* %140
store i8* null, i8** %141
store i8* null, i8** %142
%message_val113 = load %struct.message* %139
call void @enqueue(%struct.head* %138, %struct.message %message_val113)
%addr_struct114 = load %actor_address_struct** %c5
%msgQueue_ptr115 = getelementptr inbounds %actor_address_struct* %addr_struct114, i32 0, i32 2
%msgQueue116 = load i8** %msgQueue_ptr115
%143 = bitcast i8* %msgQueue116 to %struct.head*
%144 = alloca %struct.message
%malloccall117 = tail call i8* @malloc(i32 0)
%argument_struct118 = bitcast i8* %malloccall117 to %temp.17*
%case119 = alloca i32
store i32 0, i32* %case119
%145 = getelementptr inbounds %struct.message* %144, i32 0, i32 0
%146 = getelementptr inbounds %struct.message* %144, i32 0, i32 1
%147 = getelementptr inbounds %struct.message* %144, i32 0, i32 2
store i32 0, i32* %145
store i8* null, i8** %146
store i8* null, i8** %147
%message_val120 = load %struct.message* %144
call void @enqueue(%struct.head* %143, %struct.message %message_val120)
%addr_struct121 = load %actor_address_struct** %o
%msgQueue_ptr122 = getelementptr inbounds %actor_address_struct* %addr_struct121, i32 0, i32 2
%msgQueue123 = load i8** %msgQueue_ptr122
```

```
  %148 = bitcast i8* %msgQueue123 to %struct.head*
  %149 = alloca %struct.message
  %malloccall124 = tail call i8* @malloc(i32 0)
  %argument_struct125 = bitcast i8* %malloccall124 to %temp.18*
  %case126 = alloca i32
  store i32 0, i32* %case126
  %150 = getelementptr inbounds %struct.message* %149, i32 0, i32 0
  %151 = getelementptr inbounds %struct.message* %149, i32 0, i32 1
  %152 = getelementptr inbounds %struct.message* %149, i32 0, i32 2
  store i32 0, i32* %150
  store i8* null, i8** %151
  store i8* null, i8** %152
  %message_val127 = load %struct.message* %149
  call void @enqueue(%struct.head* %148, %struct.message %message_val127)
  %153 = alloca i32
  store i32 1, i32* %153
  br label %while128

while128:                                         ; preds = %while_body129, %merge85
  %"return:i132" = load i32* %153
  %tmp133 = icmp slt i32 %"return:i132", 1024
  br i1 %tmp133, label %while_body129, label %merge134

while_body129:                                    ; preds = %while128
  %idx = load i32* %153
  %pos130 = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32 %idx
  %tid_p = getelementptr inbounds %actor_address_struct* %pos130, i32 0, i32 1
  %tid_val = load i32* %tid_p
  %pthread_join_result = call i32 @pthread_join(i32 %tid_val, i8** null)
  %"return:i" = load i32* %153
  %tmp131 = add i32 %"return:i", 1
  store i32 %tmp131, i32* %153
  br label %while128

merge134:                                         ; preds = %while128
  ret i32 0
}

define i8* @caller(i8* %ptr) {
entry:
  %0 = alloca i8*
  store i8* %ptr, i8** %0
  %state_struct = alloca %caller_struct*
  %1 = load i8** %0
  %2 = bitcast i8* %1 to %caller_struct*
  store %caller_struct* %2, %caller_struct** %state_struct
```

```
  %3 = load %caller_struct** %state_struct
  %4 = getelementptr inbounds %caller_struct* %3, i32 0, i32 0
  %5 = load i32* %4
  %self_index = alloca i32
  store i32 %5, i32* %self_index
  %6 = load %caller_struct** %state_struct
  %7 = getelementptr inbounds %caller_struct* %6, i32 0, i32 1
  %8 = load %actor_address_struct** %7
  %op = alloca %actor_address_struct*
  store %actor_address_struct* %8, %actor_address_struct** %op
  %9 = load %caller_struct** %state_struct
  %10 = getelementptr inbounds %caller_struct* %9, i32 0, i32 2
  %11 = load i32* %10
  %id = alloca i32
  store i32 %11, i32* %id
  %partner = alloca %actor_address_struct*
  %partnerId = alloca i32
  %busy = alloca i32
  tail call void @free(i8* %ptr)
  store i32 9999, i32* %partnerId
  store i32 0, i32* %busy
  br label %pred_msg_while_bb

pred_msg_while_bb:                                ; preds = %entry, %finish_msg_while_bb
  br i1 true, label %body_msg_while_bb, label %merge_msg_while_bb

body_msg_while_bb:                                ; preds = %pred_msg_while_bb
  %"self:index_val" = load i32* %self_index
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%"self:index_val"
  %12 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %13 = load i8** %12
  %14 = bitcast i8* %13 to %struct.head*
  %message_struct = alloca %struct.message
  call void @dequeue(%struct.message* %message_struct, %struct.head* %14)
  %15 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 0
  %case_num = load i32* %15
  %16 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 1
  %actuals_ptr = load i8** %16
  %17 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 2
  %sender_ptr = load i8** %17
  switch i32 %case_num, label %msg_default_case_bb [
    i32 3, label %msg_cantConnect_case_bb
    i32 2, label %msg_checkIfBusy_case_bb
    i32 4, label %msg_connect_case_bb
    i32 6, label %msg_disconnect_case_bb
```

```
    i32 5, label %msg_hangUp_case_bb
    i32 1, label %msg_makeCall_case_bb
    i32 0, label %msg_die_case_bb
  ]

finish_msg_while_bb:                              ; preds = %msg_disconnect_case_bb, %merge82,
%msg_connect_case_bb, %msg_cantConnect_case_bb, %merge23, %merge, %msg_default_case_bb
  br label %pred_msg_while_bb

merge_msg_while_bb:                               ; preds = %msg_die_case_bb, %pred_msg_while_bb
  %ret = alloca i8
  ret i8* %ret

msg_die_case_bb:                                  ; preds = %body_msg_while_bb
  br label %merge_msg_while_bb

msg_makeCall_case_bb:                             ; preds = %body_msg_while_bb
  %actual_ptr = bitcast i8* %actuals_ptr to %makeCall_struct*
  %18 = alloca %makeCall_struct*
  store %makeCall_struct* %actual_ptr, %makeCall_struct** %18
  %19 = load %makeCall_struct** %18
  %f_ptr = getelementptr inbounds %makeCall_struct* %19, i32 0, i32 0
  %20 = load %actor_address_struct** %f_ptr
  %me = alloca %actor_address_struct*
  store %actor_address_struct* %20, %actor_address_struct** %me
  %f_ptr1 = getelementptr inbounds %makeCall_struct* %19, i32 0, i32 1
  %21 = load %actor_address_struct** %f_ptr1
  %dest = alloca %actor_address_struct*
  store %actor_address_struct* %21, %actor_address_struct** %dest
  %f_ptr2 = getelementptr inbounds %makeCall_struct* %19, i32 0, i32 2
  %22 = load i32* %f_ptr2
  %destId = alloca i32
  store i32 %22, i32* %destId
  %printf3 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt6, i32 0, i32
0), i8* getelementptr inbounds ([28 x i8]* @.str8, i32 0, i32 0))
  %busy4 = load i32* %busy
  %tmp = icmp eq i32 %busy4, 0
  br i1 %tmp, label %then, label %else

msg_checkIfBusy_case_bb:                          ; preds = %body_msg_while_bb
  %actual_ptr15 = bitcast i8* %actuals_ptr to %checkIfBusy_struct*
  %23 = alloca %checkIfBusy_struct*
  store %checkIfBusy_struct* %actual_ptr15, %checkIfBusy_struct** %23
  %24 = load %checkIfBusy_struct** %23
  %f_ptr16 = getelementptr inbounds %checkIfBusy_struct* %24, i32 0, i32 0
  %25 = load %actor_address_struct** %f_ptr16
```

```
  %me17 = alloca %actor_address_struct*
  store %actor_address_struct* %25, %actor_address_struct** %me17
  %f_ptr18 = getelementptr inbounds %checkIfBusy_struct* %24, i32 0, i32 1
  %26 = load %actor_address_struct** %f_ptr18
  %caller = alloca %actor_address_struct*
  store %actor_address_struct* %26, %actor_address_struct** %caller
  %f_ptr19 = getelementptr inbounds %checkIfBusy_struct* %24, i32 0, i32 2
  %27 = load i32* %f_ptr19
  %callerId = alloca i32
  store i32 %27, i32* %callerId
  %printf20 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt16, i32 0, i32
0), i8* getelementptr inbounds ([25 x i8]* @.str18, i32 0, i32 0))
  %busy21 = load i32* %busy
  %tmp22 = icmp eq i32 %busy21, 1
  br i1 %tmp22, label %then24, label %else41

msg_cantConnect_case_bb:                          ; preds = %body_msg_while_bb
  %actual_ptr60 = bitcast i8* %actuals_ptr to %cantConnect_struct*
  %28 = alloca %cantConnect_struct*
  store %cantConnect_struct* %actual_ptr60, %cantConnect_struct** %28
  %29 = load %cantConnect_struct** %28
  %f_ptr61 = getelementptr inbounds %cantConnect_struct* %29, i32 0, i32 0
  %30 = load %actor_address_struct** %f_ptr61
  %me62 = alloca %actor_address_struct*
  store %actor_address_struct* %30, %actor_address_struct** %me62
  %f_ptr63 = getelementptr inbounds %cantConnect_struct* %29, i32 0, i32 1
  %31 = load %actor_address_struct** %f_ptr63
  %dest64 = alloca %actor_address_struct*
  store %actor_address_struct* %31, %actor_address_struct** %dest64
  %printf65 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt31, i32 0, i32
0), i8* getelementptr inbounds ([25 x i8]* @.str33, i32 0, i32 0))
  store i32 0, i32* %busy
  store i32 9999, i32* %partnerId
  br label %finish_msg_while_bb

msg_connect_case_bb:                              ; preds = %body_msg_while_bb
  %actual_ptr66 = bitcast i8* %actuals_ptr to %connect_struct*
  %32 = alloca %connect_struct*
  store %connect_struct* %actual_ptr66, %connect_struct** %32
  %33 = load %connect_struct** %32
  %f_ptr67 = getelementptr inbounds %connect_struct* %33, i32 0, i32 0
  %34 = load %actor_address_struct** %f_ptr67
  %me68 = alloca %actor_address_struct*
  store %actor_address_struct* %34, %actor_address_struct** %me68
  %f_ptr69 = getelementptr inbounds %connect_struct* %33, i32 0, i32 1
  %35 = load %actor_address_struct** %f_ptr69
```

```
  %dest70 = alloca %actor_address_struct*
  store %actor_address_struct* %35, %actor_address_struct** %dest70
  %f_ptr71 = getelementptr inbounds %connect_struct* %33, i32 0, i32 2
  %36 = load i32* %f_ptr71
  %destId72 = alloca i32
  store i32 %36, i32* %destId72
  %printf73 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt36, i32 0, i32
0), i8* getelementptr inbounds ([34 x i8]* @.str38, i32 0, i32 0))
  store i32 1, i32* %busy
  %destId74 = load i32* %destId72
  store i32 %destId74, i32* %partnerId
  %dest75 = load %actor_address_struct** %dest70
  store %actor_address_struct* %dest75, %actor_address_struct** %partner
  br label %finish_msg_while_bb

msg_hangUp_case_bb:                               ; preds = %body_msg_while_bb
  %actual_ptr76 = bitcast i8* %actuals_ptr to %hangUp_struct*
  %37 = alloca %hangUp_struct*
  store %hangUp_struct* %actual_ptr76, %hangUp_struct** %37
  %38 = load %hangUp_struct** %37
  %f_ptr77 = getelementptr inbounds %hangUp_struct* %38, i32 0, i32 0
  %39 = load %actor_address_struct** %f_ptr77
  %me78 = alloca %actor_address_struct*
  store %actor_address_struct* %39, %actor_address_struct** %me78
  %printf79 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt41, i32 0, i32
0), i8* getelementptr inbounds ([27 x i8]* @.str43, i32 0, i32 0))
  %busy80 = load i32* %busy
  %tmp81 = icmp eq i32 %busy80, 1
  br i1 %tmp81, label %then83, label %else100

msg_disconnect_case_bb:                           ; preds = %body_msg_while_bb
  %actual_ptr101 = bitcast i8* %actuals_ptr to %disconnect_struct*
  %40 = alloca %disconnect_struct*
  store %disconnect_struct* %actual_ptr101, %disconnect_struct** %40
  %41 = load %disconnect_struct** %40
  %printf102 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt51, i32 0, i32
0), i8* getelementptr inbounds ([30 x i8]* @.str53, i32 0, i32 0))
  store i32 0, i32* %busy
  store i32 9999, i32* %partnerId
  %totalHangups = load i32* @totalHangups
  %tmp103 = add i32 %totalHangups, 1
  store i32 %tmp103, i32* @totalHangups
  br label %finish_msg_while_bb

msg_default_case_bb:                              ; preds = %body_msg_while_bb
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt2, i32 0, i32 0),
```

```
i8* getelementptr inbounds ([27 x i8]* @.str, i32 0, i32 0))
  br label %finish_msg_while_bb

merge:                                            ; preds = %else, %then
  br label %finish_msg_while_bb

then:                                             ; preds = %msg_makeCall_case_bb
  %printf5 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt11, i32 0, i32
0), i8* getelementptr inbounds ([31 x i8]* @.str13, i32 0, i32 0))
  store i32 1, i32* %busy
  %destId6 = load i32* %destId
  store i32 %destId6, i32* %partnerId
  %dest7 = load %actor_address_struct** %dest
  store %actor_address_struct* %dest7, %actor_address_struct** %partner
  %addr_struct = load %actor_address_struct** %op
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %42 = bitcast i8* %msgQueue to %struct.head*
  %43 = alloca %struct.message
  %partnerId8 = load i32* %partnerId
  %dest9 = load %actor_address_struct** %dest
  %id10 = load i32* %id
  %me11 = load %actor_address_struct** %me
  %malloccall = tail call i8* @malloc(i32 ptrtoint (%temp* getelementptr (%temp* null, i32 1) to
i32))
  %argument_struct = bitcast i8* %malloccall to %temp*
  %44 = alloca %actor_address_struct*
  store %actor_address_struct* %me11, %actor_address_struct** %44
  %arg_val = load %actor_address_struct** %44
  %45 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 0
  store %actor_address_struct* %arg_val, %actor_address_struct** %45
  %46 = alloca i32
  store i32 %id10, i32* %46
  %arg_val12 = load i32* %46
  %47 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 1
  store i32 %arg_val12, i32* %47
  %48 = alloca %actor_address_struct*
  store %actor_address_struct* %dest9, %actor_address_struct** %48
  %arg_val13 = load %actor_address_struct** %48
  %49 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 2
  store %actor_address_struct* %arg_val13, %actor_address_struct** %49
  %50 = alloca i32
  store i32 %partnerId8, i32* %50
  %arg_val14 = load i32* %50
  %51 = getelementptr inbounds %temp* %argument_struct, i32 0, i32 3
  store i32 %arg_val14, i32* %51
```

```
  %52 = bitcast %temp* %argument_struct to i8*
  %case = alloca i32
  store i32 7, i32* %case
  %53 = getelementptr inbounds %struct.message* %43, i32 0, i32 0
  %54 = getelementptr inbounds %struct.message* %43, i32 0, i32 1
  %55 = getelementptr inbounds %struct.message* %43, i32 0, i32 2
  store i32 7, i32* %53
  store i8* %52, i8** %54
  store i8* null, i8** %55
  %message_val = load %struct.message* %43
  call void @enqueue(%struct.head* %42, %struct.message %message_val)
  br label %merge

else:                                             ; preds = %msg_makeCall_case_bb
  br label %merge

merge23:                                          ; preds = %else41, %then24
  br label %finish_msg_while_bb

then24:                                           ; preds = %msg_checkIfBusy_case_bb
  %printf25 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt21, i32 0, i32
0), i8* getelementptr inbounds ([38 x i8]* @.str23, i32 0, i32 0))
  %addr_struct26 = load %actor_address_struct** %op
  %msgQueue_ptr27 = getelementptr inbounds %actor_address_struct* %addr_struct26, i32 0, i32 2
  %msgQueue28 = load i8** %msgQueue_ptr27
  %56 = bitcast i8* %msgQueue28 to %struct.head*
  %57 = alloca %struct.message
  %callerId29 = load i32* %callerId
  %caller30 = load %actor_address_struct** %caller
  %id31 = load i32* %id
  %me32 = load %actor_address_struct** %me17
  %malloccall33 = tail call i8* @malloc(i32 ptrtoint (%temp.0* getelementptr (%temp.0* null, i32 1)
to i32))
  %argument_struct34 = bitcast i8* %malloccall33 to %temp.0*
  %58 = alloca %actor_address_struct*
  store %actor_address_struct* %me32, %actor_address_struct** %58
  %arg_val35 = load %actor_address_struct** %58
  %59 = getelementptr inbounds %temp.0* %argument_struct34, i32 0, i32 0
  store %actor_address_struct* %arg_val35, %actor_address_struct** %59
  %60 = alloca i32
  store i32 %id31, i32* %60
  %arg_val36 = load i32* %60
  %61 = getelementptr inbounds %temp.0* %argument_struct34, i32 0, i32 1
  store i32 %arg_val36, i32* %61
  %62 = alloca %actor_address_struct*
  store %actor_address_struct* %caller30, %actor_address_struct** %62
```

```
  %arg_val37 = load %actor_address_struct** %62
  %63 = getelementptr inbounds %temp.0* %argument_struct34, i32 0, i32 2
  store %actor_address_struct* %arg_val37, %actor_address_struct** %63
  %64 = alloca i32
  store i32 %callerId29, i32* %64
  %arg_val38 = load i32* %64
  %65 = getelementptr inbounds %temp.0* %argument_struct34, i32 0, i32 3
  store i32 %arg_val38, i32* %65
  %66 = bitcast %temp.0* %argument_struct34 to i8*
  %case39 = alloca i32
  store i32 8, i32* %case39
  %67 = getelementptr inbounds %struct.message* %57, i32 0, i32 0
  %68 = getelementptr inbounds %struct.message* %57, i32 0, i32 1
  %69 = getelementptr inbounds %struct.message* %57, i32 0, i32 2
  store i32 8, i32* %67
  store i8* %66, i8** %68
  store i8* null, i8** %69
  %message_val40 = load %struct.message* %57
  call void @enqueue(%struct.head* %56, %struct.message %message_val40)
  br label %merge23

else41:                                           ; preds = %msg_checkIfBusy_case_bb
  %printf42 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt26, i32 0, i32
0), i8* getelementptr inbounds ([36 x i8]* @.str28, i32 0, i32 0))
  %addr_struct43 = load %actor_address_struct** %op
  %msgQueue_ptr44 = getelementptr inbounds %actor_address_struct* %addr_struct43, i32 0, i32 2
  %msgQueue45 = load i8** %msgQueue_ptr44
  %70 = bitcast i8* %msgQueue45 to %struct.head*
  %71 = alloca %struct.message
  %callerId46 = load i32* %callerId
  %caller47 = load %actor_address_struct** %caller
  %id48 = load i32* %id
  %me49 = load %actor_address_struct** %me17
  %malloccall50 = tail call i8* @malloc(i32 ptrtoint (%temp.1* getelementptr (%temp.1* null, i32 1)
to i32))
  %argument_struct51 = bitcast i8* %malloccall50 to %temp.1*
  %72 = alloca %actor_address_struct*
  store %actor_address_struct* %me49, %actor_address_struct** %72
  %arg_val52 = load %actor_address_struct** %72
  %73 = getelementptr inbounds %temp.1* %argument_struct51, i32 0, i32 0
  store %actor_address_struct* %arg_val52, %actor_address_struct** %73
  %74 = alloca i32
  store i32 %id48, i32* %74
  %arg_val53 = load i32* %74
  %75 = getelementptr inbounds %temp.1* %argument_struct51, i32 0, i32 1
  store i32 %arg_val53, i32* %75
```

```
  %76 = alloca %actor_address_struct*
  store %actor_address_struct* %caller47, %actor_address_struct** %76
  %arg_val54 = load %actor_address_struct** %76
  %77 = getelementptr inbounds %temp.1* %argument_struct51, i32 0, i32 2
  store %actor_address_struct* %arg_val54, %actor_address_struct** %77
  %78 = alloca i32
  store i32 %callerId46, i32* %78
  %arg_val55 = load i32* %78
  %79 = getelementptr inbounds %temp.1* %argument_struct51, i32 0, i32 3
  store i32 %arg_val55, i32* %79
  %80 = bitcast %temp.1* %argument_struct51 to i8*
  %case56 = alloca i32
  store i32 9, i32* %case56
  %81 = getelementptr inbounds %struct.message* %71, i32 0, i32 0
  %82 = getelementptr inbounds %struct.message* %71, i32 0, i32 1
  %83 = getelementptr inbounds %struct.message* %71, i32 0, i32 2
  store i32 9, i32* %81
  store i8* %80, i8** %82
  store i8* null, i8** %83
  %message_val57 = load %struct.message* %71
  call void @enqueue(%struct.head* %70, %struct.message %message_val57)
  store i32 1, i32* %busy
  %callerId58 = load i32* %callerId
  store i32 %callerId58, i32* %partnerId
  %caller59 = load %actor_address_struct** %caller
  store %actor_address_struct* %caller59, %actor_address_struct** %partner
  br label %merge23

merge82:                                          ; preds = %else100, %then83
  br label %finish_msg_while_bb

then83:                                           ; preds = %msg_hangUp_case_bb
  %printf84 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt46, i32 0, i32
0), i8* getelementptr inbounds ([30 x i8]* @.str48, i32 0, i32 0))
  store i32 0, i32* %busy
  %addr_struct85 = load %actor_address_struct** %op
  %msgQueue_ptr86 = getelementptr inbounds %actor_address_struct* %addr_struct85, i32 0, i32 2
  %msgQueue87 = load i8** %msgQueue_ptr86
  %84 = bitcast i8* %msgQueue87 to %struct.head*
  %85 = alloca %struct.message
  %partnerId88 = load i32* %partnerId
  %partner89 = load %actor_address_struct** %partner
  %id90 = load i32* %id
  %me91 = load %actor_address_struct** %me78
  %malloccall92 = tail call i8* @malloc(i32 ptrtoint (%temp.2* getelementptr (%temp.2* null, i32 1)
to i32))
```

```
  %argument_struct93 = bitcast i8* %malloccall92 to %temp.2*
  %86 = alloca %actor_address_struct*
  store %actor_address_struct* %me91, %actor_address_struct** %86
  %arg_val94 = load %actor_address_struct** %86
  %87 = getelementptr inbounds %temp.2* %argument_struct93, i32 0, i32 0
  store %actor_address_struct* %arg_val94, %actor_address_struct** %87
  %88 = alloca i32
  store i32 %id90, i32* %88
  %arg_val95 = load i32* %88
  %89 = getelementptr inbounds %temp.2* %argument_struct93, i32 0, i32 1
  store i32 %arg_val95, i32* %89
  %90 = alloca %actor_address_struct*
  store %actor_address_struct* %partner89, %actor_address_struct** %90
  %arg_val96 = load %actor_address_struct** %90
  %91 = getelementptr inbounds %temp.2* %argument_struct93, i32 0, i32 2
  store %actor_address_struct* %arg_val96, %actor_address_struct** %91
  %92 = alloca i32
  store i32 %partnerId88, i32* %92
  %arg_val97 = load i32* %92
  %93 = getelementptr inbounds %temp.2* %argument_struct93, i32 0, i32 3
  store i32 %arg_val97, i32* %93
  %94 = bitcast %temp.2* %argument_struct93 to i8*
  %case98 = alloca i32
  store i32 10, i32* %case98
  %95 = getelementptr inbounds %struct.message* %85, i32 0, i32 0
  %96 = getelementptr inbounds %struct.message* %85, i32 0, i32 1
  %97 = getelementptr inbounds %struct.message* %85, i32 0, i32 2
  store i32 10, i32* %95
  store i8* %94, i8** %96
  store i8* null, i8** %97
  %message_val99 = load %struct.message* %85
  call void @enqueue(%struct.head* %84, %struct.message %message_val99)
  store i32 9999, i32* %partnerId
  br label %merge82

else100:                                          ; preds = %msg_hangUp_case_bb
  br label %merge82
}

define i8* @operator(i8* %ptr) {
entry:
  %0 = alloca i8*
  store i8* %ptr, i8** %0
  %state_struct = alloca %operator_struct*
  %1 = load i8** %0
  %2 = bitcast i8* %1 to %operator_struct*
```

```
  store %operator_struct* %2, %operator_struct** %state_struct
  %3 = load %operator_struct** %state_struct
  %4 = getelementptr inbounds %operator_struct* %3, i32 0, i32 0
  %5 = load i32* %4
  %self_index = alloca i32
  store i32 %5, i32* %self_index
  %6 = load %operator_struct** %state_struct
  %7 = getelementptr inbounds %operator_struct* %6, i32 0, i32 1
  %8 = load i32* %7
  %capacity = alloca i32
  store i32 %8, i32* %capacity
  %count = alloca i32
  tail call void @free(i8* %ptr)
  store i32 0, i32* %count
  br label %pred_msg_while_bb

pred_msg_while_bb:                                ; preds = %entry, %finish_msg_while_bb
  br i1 true, label %body_msg_while_bb, label %merge_msg_while_bb

body_msg_while_bb:                                ; preds = %pred_msg_while_bb
  %"self:index_val" = load i32* %self_index
  %pos = getelementptr inbounds [1024 x %actor_address_struct]* @global_actors, i32 0, i32
%"self:index_val"
  %9 = getelementptr inbounds %actor_address_struct* %pos, i32 0, i32 2
  %10 = load i8** %9
  %11 = bitcast i8* %10 to %struct.head*
  %message_struct = alloca %struct.message
  call void @dequeue(%struct.message* %message_struct, %struct.head* %11)
  %12 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 0
  %case_num = load i32* %12
  %13 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 1
  %actuals_ptr = load i8** %13
  %14 = getelementptr inbounds %struct.message* %message_struct, i32 0, i32 2
  %sender_ptr = load i8** %14
  switch i32 %case_num, label %msg_default_case_bb [
    i32 0, label %msg_die_case_bb
    i32 10, label %msg_endCall_case_bb
    i32 8, label %msg_respondBusy_case_bb
    i32 9, label %msg_respondOK_case_bb
    i32 7, label %msg_tryConnect_case_bb
  ]

finish_msg_while_bb:                              ; preds = %msg_endCall_case_bb,
%msg_respondOK_case_bb, %msg_respondBusy_case_bb, %merge, %msg_default_case_bb
  br label %pred_msg_while_bb
```

```
merge_msg_while_bb:                               ; preds = %msg_die_case_bb, %pred_msg_while_bb
  %ret = alloca i8
  ret i8* %ret

msg_die_case_bb:                                  ; preds = %body_msg_while_bb
  br label %merge_msg_while_bb

msg_tryConnect_case_bb:                           ; preds = %body_msg_while_bb
  %actual_ptr = bitcast i8* %actuals_ptr to %tryConnect_struct*
  %15 = alloca %tryConnect_struct*
  store %tryConnect_struct* %actual_ptr, %tryConnect_struct** %15
  %16 = load %tryConnect_struct** %15
  %f_ptr = getelementptr inbounds %tryConnect_struct* %16, i32 0, i32 0
  %17 = load %actor_address_struct** %f_ptr
  %src = alloca %actor_address_struct*
  store %actor_address_struct* %17, %actor_address_struct** %src
  %f_ptr1 = getelementptr inbounds %tryConnect_struct* %16, i32 0, i32 1
  %18 = load i32* %f_ptr1
  %srcId = alloca i32
  store i32 %18, i32* %srcId
  %f_ptr2 = getelementptr inbounds %tryConnect_struct* %16, i32 0, i32 2
  %19 = load %actor_address_struct** %f_ptr2
  %dest = alloca %actor_address_struct*
  store %actor_address_struct* %19, %actor_address_struct** %dest
  %f_ptr3 = getelementptr inbounds %tryConnect_struct* %16, i32 0, i32 3
  %20 = load i32* %f_ptr3
  %destId = alloca i32
  store i32 %20, i32* %destId
  %printf4 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt61, i32 0, i32
0), i8* getelementptr inbounds ([38 x i8]* @.str63, i32 0, i32 0))
  %count5 = load i32* %count
  %capacity6 = load i32* %capacity
  %tmp = icmp slt i32 %count5, %capacity6
  br i1 %tmp, label %then, label %else

msg_respondBusy_case_bb:                          ; preds = %body_msg_while_bb
  %actual_ptr27 = bitcast i8* %actuals_ptr to %respondBusy_struct*
  %21 = alloca %respondBusy_struct*
  store %respondBusy_struct* %actual_ptr27, %respondBusy_struct** %21
  %22 = load %respondBusy_struct** %21
  %f_ptr28 = getelementptr inbounds %respondBusy_struct* %22, i32 0, i32 0
  %23 = load %actor_address_struct** %f_ptr28
  %dest29 = alloca %actor_address_struct*
  store %actor_address_struct* %23, %actor_address_struct** %dest29
  %f_ptr30 = getelementptr inbounds %respondBusy_struct* %22, i32 0, i32 1
  %24 = load i32* %f_ptr30
```

```
  %destId31 = alloca i32
  store i32 %24, i32* %destId31
  %f_ptr32 = getelementptr inbounds %respondBusy_struct* %22, i32 0, i32 2
  %25 = load %actor_address_struct** %f_ptr32
  %src33 = alloca %actor_address_struct*
  store %actor_address_struct* %25, %actor_address_struct** %src33
  %f_ptr34 = getelementptr inbounds %respondBusy_struct* %22, i32 0, i32 3
  %26 = load i32* %f_ptr34
  %srcId35 = alloca i32
  store i32 %26, i32* %srcId35
  %printf36 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt76, i32 0, i32
0), i8* getelementptr inbounds ([24 x i8]* @.str78, i32 0, i32 0))
  %addr_struct37 = load %actor_address_struct** %src33
  %msgQueue_ptr38 = getelementptr inbounds %actor_address_struct* %addr_struct37, i32 0, i32 2
  %msgQueue39 = load i8** %msgQueue_ptr38
  %27 = bitcast i8* %msgQueue39 to %struct.head*
  %28 = alloca %struct.message
  %dest40 = load %actor_address_struct** %dest29
  %src41 = load %actor_address_struct** %src33
  %malloccall42 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct43 = bitcast i8* %malloccall42 to %temp.5*
  %29 = alloca %actor_address_struct*
  store %actor_address_struct* %src41, %actor_address_struct** %29
  %arg_val44 = load %actor_address_struct** %29
  %30 = getelementptr inbounds %temp.5* %argument_struct43, i32 0, i32 0
  store %actor_address_struct* %arg_val44, %actor_address_struct** %30
  %31 = alloca %actor_address_struct*
  store %actor_address_struct* %dest40, %actor_address_struct** %31
  %arg_val45 = load %actor_address_struct** %31
  %32 = getelementptr inbounds %temp.5* %argument_struct43, i32 0, i32 1
  store %actor_address_struct* %arg_val45, %actor_address_struct** %32
  %33 = bitcast %temp.5* %argument_struct43 to i8*
  %case46 = alloca i32
  store i32 3, i32* %case46
  %34 = getelementptr inbounds %struct.message* %28, i32 0, i32 0
  %35 = getelementptr inbounds %struct.message* %28, i32 0, i32 1
  %36 = getelementptr inbounds %struct.message* %28, i32 0, i32 2
  store i32 3, i32* %34
  store i8* %33, i8** %35
  store i8* null, i8** %36
  %message_val47 = load %struct.message* %28
  call void @enqueue(%struct.head* %27, %struct.message %message_val47)
  %count48 = load i32* %count
  %tmp49 = sub i32 %count48, 1
  store i32 %tmp49, i32* %count
```

```
  br label %finish_msg_while_bb

msg_respondOK_case_bb:                            ; preds = %body_msg_while_bb
  %actual_ptr50 = bitcast i8* %actuals_ptr to %respondOK_struct*
  %37 = alloca %respondOK_struct*
  store %respondOK_struct* %actual_ptr50, %respondOK_struct** %37
  %38 = load %respondOK_struct** %37
  %f_ptr51 = getelementptr inbounds %respondOK_struct* %38, i32 0, i32 0
  %39 = load %actor_address_struct** %f_ptr51
  %dest52 = alloca %actor_address_struct*
  store %actor_address_struct* %39, %actor_address_struct** %dest52
  %f_ptr53 = getelementptr inbounds %respondOK_struct* %38, i32 0, i32 1
  %40 = load i32* %f_ptr53
  %destId54 = alloca i32
  store i32 %40, i32* %destId54
  %f_ptr55 = getelementptr inbounds %respondOK_struct* %38, i32 0, i32 2
  %41 = load %actor_address_struct** %f_ptr55
  %src56 = alloca %actor_address_struct*
  store %actor_address_struct* %41, %actor_address_struct** %src56
  %f_ptr57 = getelementptr inbounds %respondOK_struct* %38, i32 0, i32 3
  %42 = load i32* %f_ptr57
  %srcId58 = alloca i32
  store i32 %42, i32* %srcId58
  %printf59 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt81, i32 0, i32
0), i8* getelementptr inbounds ([30 x i8]* @.str83, i32 0, i32 0))
  %addr_struct60 = load %actor_address_struct** %src56
  %msgQueue_ptr61 = getelementptr inbounds %actor_address_struct* %addr_struct60, i32 0, i32 2
  %msgQueue62 = load i8** %msgQueue_ptr61
  %43 = bitcast i8* %msgQueue62 to %struct.head*
  %44 = alloca %struct.message
  %destId63 = load i32* %destId54
  %dest64 = load %actor_address_struct** %dest52
  %src65 = load %actor_address_struct** %src56
  %malloccall66 = tail call i8* @malloc(i32 ptrtoint (%temp.6* getelementptr (%temp.6* null, i32 1)
to i32))
  %argument_struct67 = bitcast i8* %malloccall66 to %temp.6*
  %45 = alloca %actor_address_struct*
  store %actor_address_struct* %src65, %actor_address_struct** %45
  %arg_val68 = load %actor_address_struct** %45
  %46 = getelementptr inbounds %temp.6* %argument_struct67, i32 0, i32 0
  store %actor_address_struct* %arg_val68, %actor_address_struct** %46
  %47 = alloca %actor_address_struct*
  store %actor_address_struct* %dest64, %actor_address_struct** %47
  %arg_val69 = load %actor_address_struct** %47
  %48 = getelementptr inbounds %temp.6* %argument_struct67, i32 0, i32 1
  store %actor_address_struct* %arg_val69, %actor_address_struct** %48
```

```
  %49 = alloca i32
  store i32 %destId63, i32* %49
  %arg_val70 = load i32* %49
  %50 = getelementptr inbounds %temp.6* %argument_struct67, i32 0, i32 2
  store i32 %arg_val70, i32* %50
  %51 = bitcast %temp.6* %argument_struct67 to i8*
  %case71 = alloca i32
  store i32 4, i32* %case71
  %52 = getelementptr inbounds %struct.message* %44, i32 0, i32 0
  %53 = getelementptr inbounds %struct.message* %44, i32 0, i32 1
  %54 = getelementptr inbounds %struct.message* %44, i32 0, i32 2
  store i32 4, i32* %52
  store i8* %51, i8** %53
  store i8* null, i8** %54
  %message_val72 = load %struct.message* %44
  call void @enqueue(%struct.head* %43, %struct.message %message_val72)
  br label %finish_msg_while_bb

msg_endCall_case_bb:                              ; preds = %body_msg_while_bb
  %actual_ptr73 = bitcast i8* %actuals_ptr to %endCall_struct*
  %55 = alloca %endCall_struct*
  store %endCall_struct* %actual_ptr73, %endCall_struct** %55
  %56 = load %endCall_struct** %55
  %f_ptr74 = getelementptr inbounds %endCall_struct* %56, i32 0, i32 0
  %57 = load %actor_address_struct** %f_ptr74
  %src75 = alloca %actor_address_struct*
  store %actor_address_struct* %57, %actor_address_struct** %src75
  %f_ptr76 = getelementptr inbounds %endCall_struct* %56, i32 0, i32 1
  %58 = load i32* %f_ptr76
  %srcId77 = alloca i32
  store i32 %58, i32* %srcId77
  %f_ptr78 = getelementptr inbounds %endCall_struct* %56, i32 0, i32 2
  %59 = load %actor_address_struct** %f_ptr78
  %dest79 = alloca %actor_address_struct*
  store %actor_address_struct* %59, %actor_address_struct** %dest79
  %f_ptr80 = getelementptr inbounds %endCall_struct* %56, i32 0, i32 3
  %60 = load i32* %f_ptr80
  %destId81 = alloca i32
  store i32 %60, i32* %destId81
  %printf82 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt86, i32 0, i32
0), i8* getelementptr inbounds ([31 x i8]* @.str88, i32 0, i32 0))
  %addr_struct83 = load %actor_address_struct** %dest79
  %msgQueue_ptr84 = getelementptr inbounds %actor_address_struct* %addr_struct83, i32 0, i32 2
  %msgQueue85 = load i8** %msgQueue_ptr84
  %61 = bitcast i8* %msgQueue85 to %struct.head*
  %62 = alloca %struct.message
```

```
  %malloccall86 = tail call i8* @malloc(i32 0)
  %argument_struct87 = bitcast i8* %malloccall86 to %temp.7*
  %case88 = alloca i32
  store i32 6, i32* %case88
  %63 = getelementptr inbounds %struct.message* %62, i32 0, i32 0
  %64 = getelementptr inbounds %struct.message* %62, i32 0, i32 1
  %65 = getelementptr inbounds %struct.message* %62, i32 0, i32 2
  store i32 6, i32* %63
  store i8* null, i8** %64
  store i8* null, i8** %65
  %message_val89 = load %struct.message* %62
  call void @enqueue(%struct.head* %61, %struct.message %message_val89)
  %count90 = load i32* %count
  %tmp91 = add i32 %count90, 1
  store i32 %tmp91, i32* %count
  br label %finish_msg_while_bb

msg_default_case_bb:                              ; preds = %body_msg_while_bb
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt56, i32 0, i32
0), i8* getelementptr inbounds ([27 x i8]* @.str58, i32 0, i32 0))
  br label %finish_msg_while_bb

merge:                                            ; preds = %else, %then
  br label %finish_msg_while_bb

then:                                             ; preds = %msg_tryConnect_case_bb
  %printf7 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt66, i32 0, i32
0), i8* getelementptr inbounds ([52 x i8]* @.str68, i32 0, i32 0))
  %addr_struct = load %actor_address_struct** %dest
  %msgQueue_ptr = getelementptr inbounds %actor_address_struct* %addr_struct, i32 0, i32 2
  %msgQueue = load i8** %msgQueue_ptr
  %66 = bitcast i8* %msgQueue to %struct.head*
  %67 = alloca %struct.message
  %srcId8 = load i32* %srcId
  %src9 = load %actor_address_struct** %src
  %dest10 = load %actor_address_struct** %dest
  %malloccall = tail call i8* @malloc(i32 ptrtoint (%temp.3* getelementptr (%temp.3* null, i32 1) to
i32))
  %argument_struct = bitcast i8* %malloccall to %temp.3*
  %68 = alloca %actor_address_struct*
  store %actor_address_struct* %dest10, %actor_address_struct** %68
  %arg_val = load %actor_address_struct** %68
  %69 = getelementptr inbounds %temp.3* %argument_struct, i32 0, i32 0
  store %actor_address_struct* %arg_val, %actor_address_struct** %69
  %70 = alloca %actor_address_struct*
  store %actor_address_struct* %src9, %actor_address_struct** %70
```

```
  %arg_val11 = load %actor_address_struct** %70
  %71 = getelementptr inbounds %temp.3* %argument_struct, i32 0, i32 1
  store %actor_address_struct* %arg_val11, %actor_address_struct** %71
  %72 = alloca i32
  store i32 %srcId8, i32* %72
  %arg_val12 = load i32* %72
  %73 = getelementptr inbounds %temp.3* %argument_struct, i32 0, i32 2
  store i32 %arg_val12, i32* %73
  %74 = bitcast %temp.3* %argument_struct to i8*
  %case = alloca i32
  store i32 2, i32* %case
  %75 = getelementptr inbounds %struct.message* %67, i32 0, i32 0
  %76 = getelementptr inbounds %struct.message* %67, i32 0, i32 1
  %77 = getelementptr inbounds %struct.message* %67, i32 0, i32 2
  store i32 2, i32* %75
  store i8* %74, i8** %76
  store i8* null, i8** %77
  %message_val = load %struct.message* %67
  call void @enqueue(%struct.head* %66, %struct.message %message_val)
  %count13 = load i32* %count
  %tmp14 = add i32 %count13, 1
  store i32 %tmp14, i32* %count
  br label %merge

else:                                             ; preds = %msg_tryConnect_case_bb
  %printf15 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt71, i32 0, i32
0), i8* getelementptr inbounds ([31 x i8]* @.str73, i32 0, i32 0))
  %addr_struct16 = load %actor_address_struct** %src
  %msgQueue_ptr17 = getelementptr inbounds %actor_address_struct* %addr_struct16, i32 0, i32 2
  %msgQueue18 = load i8** %msgQueue_ptr17
  %78 = bitcast i8* %msgQueue18 to %struct.head*
  %79 = alloca %struct.message
  %dest19 = load %actor_address_struct** %dest
  %src20 = load %actor_address_struct** %src
  %malloccall21 = tail call i8* @malloc(i32 trunc (i64 mul nuw (i64 ptrtoint (i1** getelementptr
(i1** null, i32 1) to i64), i64 2) to i32))
  %argument_struct22 = bitcast i8* %malloccall21 to %temp.4*
  %80 = alloca %actor_address_struct*
  store %actor_address_struct* %src20, %actor_address_struct** %80
  %arg_val23 = load %actor_address_struct** %80
  %81 = getelementptr inbounds %temp.4* %argument_struct22, i32 0, i32 0
  store %actor_address_struct* %arg_val23, %actor_address_struct** %81
  %82 = alloca %actor_address_struct*
  store %actor_address_struct* %dest19, %actor_address_struct** %82
  %arg_val24 = load %actor_address_struct** %82
  %83 = getelementptr inbounds %temp.4* %argument_struct22, i32 0, i32 1
```

```
  store %actor_address_struct* %arg_val24, %actor_address_struct** %83
  %84 = bitcast %temp.4* %argument_struct22 to i8*
  %case25 = alloca i32
  store i32 3, i32* %case25
  %85 = getelementptr inbounds %struct.message* %79, i32 0, i32 0
  %86 = getelementptr inbounds %struct.message* %79, i32 0, i32 1
  %87 = getelementptr inbounds %struct.message* %79, i32 0, i32 2
  store i32 3, i32* %85
  store i8* %84, i8** %86
  store i8* null, i8** %87
  %message_val26 = load %struct.message* %79
  call void @enqueue(%struct.head* %78, %struct.message %message_val26)
  br label %merge
}

declare void @free(i8*)

declare noalias i8* @malloc(i32)
```

Test suites

As we added features to our language we added tests to test each one. We generally tried to stick to writing one pass and one fail test. We found that to be the best way to add new features - add it and get it completely working end-to-end, and write a test at the same time in order to verify that it's working properly. A copy of our test script is in the appendix, and a sample output of our test listing is here:

```
test-actor-all-msg-working...OK
test-actor-msg-formal-scope1...OK
test-actor-msg-formal-scope2...OK
test-actor-msg-formal-vars...OK
test-actor-msg-local-vars...OK
test-actors-syntax...OK
test-ar-binops-float.th...OK
test-ar-binops-int.th...OK
test-ar-binops.th...OK
test-char-lit-escape...OK
test-char-lit...OK
test-char-var-escape...OK
test-char-var...OK
test-comparison-float-vars...OK
```

```
test-comparison-float...OK
test-comparison-int-vars...OK
test-comparison-int...OK
test-for...OK
test-func1...OK
test-func2...OK
test-function-locals...OK
test-geturl...OK
test-hello...OK
test-ifelse...OK
test-interleaved-decls...OK
test-interleaving-global...OK
test-list1...OK
test-logical-ops...OK
test-msg-can-change-actor-formal-vars...OK
test-msg-can-change-actor-local-vars...OK
test-msg-shadowing-formal-var...OK
test-msg-shadowing-local-var...OK
test-multi-line-comments...OK
test-new-actor-in-actor-msgs...OK
test-new-actor-in-actor...OK
test-new-actor-syntax...OK
test-new-actor-w-statements.th...OK
test-print...OK
test-send-syntax...OK
test-single-line-comments...OK
test-string-lit-escape...OK
test-string-var-escape...OK
test-string-vars...OK
test-var-def-decl...OK
test-while...OK
fail-actor-msg-formal-vars1...OK
fail-actor-msg-formal-vars2...OK
fail-actor-msg-formal-vars3...OK
fail-actor-msg-local-vars1...OK
fail-actor-msg-local-vars2...OK
fail-assign1...OK
fail-assign2...OK
fail-dead1...OK
fail-expr1...OK
fail-expr2...OK
fail-for1...OK
fail-for2...OK
fail-for3...OK
fail-for4...OK
fail-for5...OK
```

```
fail-func...OK
fail-func2...OK
fail-func3...OK
fail-func4...OK
fail-interleaved-decls...OK
fail-interleaved-redecl...OK
fail-list1...OK
```

Test automation

We used a test running script, testall.sh, provided in the appendix, to automate the running of our 60+ tests.

Lessons Learned

Suraj :

1. We encountered a few unexpected behavior of our code towards the very end of our project deadline. Lesson learned: write more tests
2. It's a good idea to start messing with the microC code as early as possible. You learn when you break it and then fix it.

Linda :

1. When trying to debug LLVM code, insert `raise(Failure(L.string_of_llvm var_name))` to print out the actual llvm instruction that's being created. For example, this could inform you that `var_name` is actually a llvalue and not a pointer to an llvalue, as you had expected. This was the main culprit of a lot of our llvm errors.
2. Name llvm variables with unique/useful names. If there's an issue with an LLVM instruction, LLVM will usually print out the instruction causing the problem. Good variable names will make debugging errors like these easier because you'll more easily identify the variable name being used.
3. Due to the nature of our language (not throwing many compile-time errors and that we convert actors into while loops), our focus was on codegen and we did not flesh out semant very well. We then had a hard time debugging the llvm errors that were being thrown. Since we have a let it crash philosophy, we had to address a lot of these errors in codegen. If we had more time, I would have liked to see more tests that addresses bad

inputs and other bad programming aspects that in another language would have thrown a compile time error. Lesson: do semantic checking!

4. Try to address warnings as they come up. It's hard to understand your own code, let alone someone else's, after some time and you'll have a hard time addressing the warnings.
5. Pair programming the codegen portion of the project will save you time and might just save your sanity.
6. Write out the set of git commands and procedure for committing changes. This will ensure that everyone is committing the same way.
7. We were able to be very productive through Google Hangouts (all of us lived off campus), so don't feel pressured to always meet in person.
8. If you're unsure how to convert your cool, new language feature into LLVM, write a C program that simulates the behavior your want. Then run clang on the .c file to convert it into a .ll file with LLVM instructions. This will give you a starting point on which Ocaml LLVM IR instructions you need to be calling.
9. We divided a lot of the work by big features (i.e. actor-while loops, queues, and switch-cases) that team members owned and debugged. This was more useful than someone owning a file (mainly because the meat of our program occurred in codegen). BUT we were nervous about how the features would actually integrate. If you're in a similar scenario, we found it useful to do integration all together with one person driving. We then divided up any problems that arose among the team members. Lesson: pair (group?) program the integration of features.

Mike:

1. Coordinate your source control and have everyone on the same page about the process for making and pushing changes to the project. The project is a great opportunity to get hands on experience source control with other team members.
2. Keep up with your semantic checking as you add to your language - it might seem like a chore and it's easy to push it off when you're rushing to add that next major cool feature and you just want it to work… but it will produce dividends as the project gets bigger. Especially because you may find yourself in a situation where you're writing demos more complex than your tests have been up to that point because you're excited about that new feature and want to push it to its limits, and you're running into segfaults and freaking out that your feature is broken - it isn't, it's just that you wrote a semantically borked program in your own language, dummy.

Betsy

1. Agree with Linda about google hangouts. Towards the end we would just work together virtually with a hang out always on. Two people would be pair programming and sharing a screen with all 4 of us, while the other two might be listening while also working on something else silently. I gained a lot from this from pairing, as well as from just passively watching other people pair or solo program via screenshare. Working this way

was also nice for camaraderie morale, and because we could easily voice an issue we might be having and one or more people would be ready and willing to help right away.

2. Don't be afraid to be transparent with your teammates if you don't understand something or if you want to know more about how they implemented anything. You are all in this together and it's in their interests to have everyone be as informed as possible as well. On that note, make sure to be the kind of team member who is approachable and makes the other members feel comfortable asking for anything as well.
3. If you feel overwhelmed and like you are not exactly excited about the project because you won't be able to pull it off, it just means you need to do more and once you are actually into the thick of it, you will be SO excited about it! So don't worry if you feel daunted (like I did), just keep pushing through and soon the fear will be replaced with enthusiasm!
4. Making our language was an amazing, fun, thrilling experience. But I imagine it would have been really painful and difficult had I not been blessed with three kind, hardworking, fun people. So I think the most important advice is obvious, but it's to be a good team member for the sake of your team. We were all strangers to each other before and met on Piazza, so don't think that not having friends in the class going in lowers your chances of having a super enjoyable and rewarding group project experience.

Appendix

scanner.mll

```
(* Ocamllex scanner for Theatr

Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*)

{
    open Parser

    let make_char c =
        let ec = Char.escaped c in
        let s = Scanf.unescaped (match String.length ec with
            1 -> "\\" ^ ec
          | _ -> ec) in
```

```
        String.get s 0
}

let digit = ['0'-'9']
let double = ('-'?)((digit+'.'digit*) | ('.'digit+))
let simp_chr = [' '-'!' '#'-'&' '('-'[' ']'-'~']


let esc_chr = ['t' 'n' 'r' '\'' '\"' '\\']

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| "//"     { single_line_comment lexbuf }
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| "receive" { RECEIVE }
| "drop"   { DROP }
| "after"  { AFTER }
| "->"     { FUNC_ARROW }
| "func"   { FUNC_DECL }
| "new"    { NEW }
| "actor"  { ACTOR }
| "struct" { STRUCT }
| '.'      { DOT }
| '|'      { PIPE }
| ':'      { COLON }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| "if"     { IF }
```

```
| "else"   { ELSE }
| "for"    { FOR }
| "list"   { LIST }
| "array"  { ARRAY }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
| "double" { DOUBLE }
| "bool"   { BOOL }
| "void"   { VOID }
| "true"   { TRUE }
| "false"  { FALSE }
| "char"   { CHAR }
| "string" { STRING }

(* Literals *)
| digit+ as lxm { INTLIT(int_of_string lxm) }
| double as lxm { DOUBLELIT(float_of_string lxm)}
| '\'' (simp_chr as lxm) '\'' { CHARLIT(lxm) }
| "'\\" (esc_chr as ec) "'" { CHARLIT(make_char ec) }
| '"'      { read_string (Buffer.create 17) lexbuf }

(* Identifiers *)
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }

| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

(* From: realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html *)
and read_string buf =
  parse
  | '"'       { STRINGLIT (Buffer.contents buf) }
  | '\\' '/'  { Buffer.add_char buf '/'; read_string buf lexbuf }
  | '\\' '\\' { Buffer.add_char buf '\\'; read_string buf lexbuf }
  | '\\' 'b'  { Buffer.add_char buf '\b'; read_string buf lexbuf }
  | '\\' 'f'  { Buffer.add_char buf '\012'; read_string buf lexbuf }
  | '\\' 'n'  { Buffer.add_char buf '\n'; read_string buf lexbuf }
  | '\\' 'r'  { Buffer.add_char buf '\r'; read_string buf lexbuf }
  | '\\' 't'  { Buffer.add_char buf '\t'; read_string buf lexbuf }
  | '\\' '\"' { Buffer.add_char buf '\"'; read_string buf lexbuf }
  | [^ '"' '\\']+
    { Buffer.add_string buf (Lexing.lexeme lexbuf);
      read_string buf lexbuf
    }

  | _ as char { raise (Failure("illegal string character: " ^ Lexing.lexeme lexbuf))
```

```
}
  | eof { raise (Failure("String is not terminated")) }


and comment = parse
  "*/" { token lexbuf }
| _    { comment lexbuf }

and single_line_comment = parse
| '\n'  { token lexbuf }
| _     { single_line_comment lexbuf }
```

parser.mly

```
/* Ocamlyacc parser for Theatr
Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*/

%{
open Ast
%}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA COLON LBRACKET RBRACKET
%token FUNC_DECL FUNC_ARROW
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token RETURN IF ELSE FOR WHILE INT DOUBLE BOOL VOID STRING CHAR
%token RECEIVE DROP AFTER
%token NEW ACTOR STRUCT DOT
%token PIPE
%token LIST ARRAY STRING
%token <int> INTLIT
%token <float> DOUBLELIT
%token <string> STRINGLIT
%token <char> CHARLIT
%token <string> ID
%token EOF
```

```
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%right NOT NEG

%start program
%type <Ast.program> program

%%

program:
  decls EOF { $1 }

decls:
   /* nothing */ { [], [], [] }
 | decls vdecl { match $1 with (vars,funcs,actors) -> (($2 :: vars), funcs, actors)
}
 | decls fdecl { match $1 with (vars,funcs,actors) -> (vars, ($2 :: funcs), actors)
}
 | decls adecl { match $1 with (vars,funcs,actors) -> (vars, funcs, ($2 :: actors))
}

/* actor declaration
LBRACE and RBRACE: these will be inserted by the preprocessor
formals_opt and vdecl_list act the same as in fdecl
receives is a msg_decl list
rop and after are stmt lists
*/
adecl:
   ID LPAREN formals_opt RPAREN COLON LBRACE stmt_list receives drop after RBRACE
     { { aname = $1;
        aformals = $3;
        alocals = List.rev $7;
        receives = $8;
        drop = $9;
        after = $10; } }

receives:
    /* nothing */ { [] }
```

```
  | RECEIVE COLON LBRACE msg_decl_list RBRACE { List.rev $4 }

msg_decl_list:
    /* nothing */ { [] }
  | msg_decl_list msg_decl { $2 :: $1 }

/* declaration of what an actor does upon receiving the message
structure is very similar to functions
*/
msg_decl:
    ID LPAREN formals_opt RPAREN COLON LBRACE stmt_list RBRACE
      { { mname = $1;
         mformals = $3;
         mbody = List.rev $7 } }

drop:
      /* nothing */ { { dabody = [] } }
    | DROP COLON LBRACE stmt_list RBRACE
      { { dabody = List.rev $4 } }

after:
      /* nothing */ { { dabody = [] } }
    | AFTER COLON LBRACE stmt_list RBRACE
      { { dabody = List.rev $4 } }

fdecl:
   FUNC_DECL ID LPAREN formals_opt RPAREN FUNC_ARROW typ COLON LBRACE stmt_list
RBRACE
     { { typ = $7;
        fname = $2;
        formals = $4;
        body = List.rev $10 } }

formals_opt:
    /* nothing */ { [] }
  | formal_list   { List.rev $1 }

formal_list:
    typ ID                   { [($1,$2)] }
  | formal_list COMMA typ ID { ($3,$4) :: $1 }

ptyp:
    INT { Int }
  | DOUBLE { Double }
  | BOOL { Bool }
  | VOID { Void }
```

```
  | ACTOR { Actor }
  | STRING { String }
  | CHAR { Char }
ctyp:
    LIST { List }
  | ARRAY { Array }

typ:
    ptyp { Ptyp($1) }
  | ctyp LT ptyp GT { Ctyp($1, $3) }

vdecl:
   typ ID SEMI { ($1, $2) }

stmt_list:
    /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
    expr SEMI { Expr $1 }
  | typ ID SEMI   { Vdecl($1, $2) }
  | typ ID ASSIGN expr SEMI { Vdef({ vtyp = $1; vname = $2; vvalue = Assign($2, $4)
}) }
  | RETURN SEMI { Return Noexpr }
  | RETURN expr SEMI { Return $2 }
  | LBRACE stmt_list RBRACE { Block(List.rev $2) }
  | IF LPAREN expr RPAREN COLON stmt %prec NOELSE { If($3, $6, Block([])) }
  | IF LPAREN expr RPAREN COLON stmt ELSE COLON stmt    { If($3, $6, $9) }
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN COLON stmt
     { For($3, $5, $7, $10) }
  | WHILE LPAREN expr RPAREN COLON stmt { While($3, $6) }

expr_opt:
    /* nothing */ { Noexpr }
  | expr          { $1 }

expr:
    INTLIT          { IntLit($1) }
  | DOUBLELIT        { DoubleLit($1) }
  | STRINGLIT        { StringLit($1) }
  | CHARLIT          { CharLit($1) }
  | TRUE             { BoolLit(true) }
  | FALSE            { BoolLit(false) }
  | ID               { Id($1) }
  | LIST LBRACKET actuals_opt RBRACKET { ListC($3) }
  | ARRAY LBRACKET actuals_opt RBRACKET { ArrayC($3) }
```

```
  | expr PLUS   expr { Binop($1, Add,   $3) }
  | expr MINUS  expr { Binop($1, Sub,   $3) }
  | expr TIMES  expr { Binop($1, Mult,  $3) }
  | expr DIVIDE expr { Binop($1, Div,   $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LEQ    expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3) }
  | expr AND    expr { Binop($1, And,   $3) }
  | expr OR     expr { Binop($1, Or,    $3) }
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | NOT expr         { Unop(Not, $2) }
  | ID ASSIGN expr   { Assign($1, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }
  | NEW ID LPAREN actuals_opt RPAREN { NewActor($2, $4) }
  | ID DOT ID LPAREN actuals_opt RPAREN PIPE ID { Send($1, $3, $5, $8) }

actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                    { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

ast.ml

```
(* Abstract Syntax Tree and functions for printing it

Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*)
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq |
          And | Or

type uop = Neg | Not
```

```
type ptyp = Int | Bool | Double | Void | Actor | String | Char

type ctyp = List | Array

type typ = Ptyp of ptyp | Ctyp of ctyp * ptyp

type bind = typ * string

type sdecl = {
    name : string;
    elements : bind list
  }

type expr =
    IntLit of int
  | DoubleLit of float
  | StringLit of string
  | CharLit of char
  | BoolLit of bool
  | ListC of expr list
  | ArrayC of expr list
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of string * expr
  | Call of string * expr list
  | NewActor of string * expr list
  | Send of string * string * expr list * string
  | Noexpr

type vdef = {
    vtyp : typ;
    vname : string;
    vvalue : expr;
  }

type sdef  = {
    sname : string;
    styp : string;
    svalue : vdef list;
  }

type stmt =
    Block of stmt list
  | Expr of expr
  | Vdecl of bind
```

```
  | Vdef of vdef
  | Sdef of sdef
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind list;
    body : stmt list;
  }

type msg_decl = {
    mname : string;
    mformals : bind list;
    mbody : stmt list;
  }

type drop_after_decl = {
    dabody : stmt list;
  }

type actor_decl = {
    aname : string;
    aformals : bind list;
    alocals : stmt list;
    receives : msg_decl list;
    drop : drop_after_decl;
    after : drop_after_decl;
  }

type program = bind list * func_decl list * actor_decl list

(* Pretty-printing functions *)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
```

```
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"

let string_of_uop = function
    Neg -> "-"
  | Not -> "!"

let rec string_of_expr = function
    IntLit(l) -> string_of_int l
  | DoubleLit(f) -> string_of_float f
  | StringLit(s) -> s
  | CharLit(c)   -> Char.escaped c
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | Id(s) -> s
  | ArrayC(el) -> "Array" ^ "[" ^ String.concat ", " (List.map string_of_expr el) ^
"]"
  | ListC(el) -> "List" ^ "[" ^ String.concat ", " (List.map string_of_expr el) ^
"]"
  | 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
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | NewActor(a, el) ->
     "new" ^ a ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""
  | Send(aTyp, mname, el, a) ->
         aTyp ^ "." ^ mname ^ String.concat ", " (List.map string_of_expr el) ^ ")
| " ^ a

let string_of_ptyp = function
    Int -> "int"
  | Bool -> "bool"
  | Void -> "void"
  | Actor -> "actor"
  | Double -> "double"
  | String -> "string"
  | Char    -> "char"

let string_of_ctyp = function
    List -> "list"
  | Array -> "array"
```

```
let string_of_typ = function
    Ptyp(e) -> string_of_ptyp e
  | Ctyp(c, e) -> string_of_ctyp c ^ "<" ^ string_of_ptyp e ^ ">"

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id

let string_of_vdef vdef =
  string_of_typ vdef.vtyp ^ " " ^ vdef.vname ^ " = " ^ string_of_expr vdef.vvalue

let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n"
  | Vdecl((t, id)) -> string_of_vdecl (t, id) ^ "\n";
  | Vdef(vdef) -> string_of_vdef vdef ^ ";\n"
  | Sdef(sdef) -> "struct " ^ sdef.sname ^ " = new " ^ sdef.styp ^ "()\n"
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_mdecl mdecl =
  mdecl.mname ^ "(" ^ String.concat ", " (List.map snd mdecl.mformals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_stmt mdecl.mbody) ^
  "}\n"

let string_of_adecl adecl =
  adecl.aname ^ "(" ^ String.concat ", " (List.map snd adecl.aformals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_stmt adecl.alocals) ^
  "\nreceives:\n" ^ String.concat "" (List.map string_of_mdecl adecl.receives) ^
```

```
  "\ndrop:\n" ^ String.concat "" (List.map string_of_stmt adecl.drop.dabody) ^
  "\nafter:\n" ^ String.concat "" (List.map string_of_stmt adecl.after.dabody) ^
  "}\n"

let string_of_sdecl sdecl =
  "struct " ^ sdecl.name ^ " {\n" ^  String.concat "\n" (List.map string_of_vdecl
sdecl.elements) ^ "\n}"

let string_of_program (vars, funcs, actors) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs) ^ "\n" ^
  String.concat "\n" (List.map string_of_adecl actors) ^ "\n"
```

semant.ml

```
(* Semantic checking for the Theatr compiler

Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*)

open Ast

module StringMap = Map.Make(String)

(* Semantic checking of a program. Returns void if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each function *)

let check (globals, functions, actors) =

  (* Raise an exception if the given list has a duplicate *)
  let report_duplicate exceptf list =
    let rec helper = function
       n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
      | _ :: t -> helper t
      | [] -> ()
    in helper (List.sort compare list)
  in
```

```
(* Raise an exception if a given binding is to a void type *)
let check_not_void exceptf = function
    (Ptyp(Void), n) -> raise (Failure (exceptf n))
  | _ -> ()
in

(* Raise an exception of the given rvalue type cannot be assigned to
   the given lvalue type *)
let check_assign lvaluet rvaluet err =
   if lvaluet = rvaluet then lvaluet else raise err
in

(* Dynamically adds to MapString using list of tuples *)
let rec add_to_map m li =
  match li with
  | [] -> m
  | hd :: tl -> add_to_map (StringMap.add (fst hd) (snd hd) m) tl
in

(**** Checking Actors ****)
report_duplicate (fun n -> "duplicate actor " ^ n)
  (List.map (fun an -> an.aname) actors);

let actor_decls = List.fold_left (fun m ad -> StringMap.add ad.aname ad m)
                                 StringMap.empty actors
in

let actor_decl s = try StringMap.find s actor_decls
      with Not_found -> raise (Failure ("unrecognized actor " ^ s))
in

(**** Checking Global Variables ****)

List.iter (check_not_void (fun n -> "illegal void global " ^ n)) globals;

report_duplicate (fun n -> "duplicate global " ^ n) (List.map snd globals);

(**** Checking Functions ****)

if List.mem "print" (List.map (fun fd -> fd.fname) functions)
then raise (Failure ("function print may not be defined")) else ();

report_duplicate (fun n -> "duplicate function " ^ n)
  (List.map (fun fd -> fd.fname) functions);
```

```
  (* Formats list of different print decls by type *)
  let format_print_decls typ = ("print:" ^
    string_of_typ typ,
    {typ = Ptyp(Void); fname = "print_" ^ string_of_typ typ; formals = [(typ, "x")];
body = []})
  in

  let print_typs = List.map format_print_decls [Ptyp(String); Ptyp(Char); Ptyp(Int);
Ptyp(Double); Ptyp(Bool)] in

  let geturl = ("geturl", {typ = Ptyp(Int); fname = "geturl"; formals =
[(Ptyp(String), "x"); (Ptyp(String), "y")]; body=[]}) in

  let built_in_print_decls = add_to_map StringMap.empty print_typs in
  let built_in_decls = add_to_map built_in_print_decls [geturl] in

  (* Function declarations for named functions  *)
  let function_decls = List.fold_left (fun m fd -> StringMap.add fd.fname fd m)
                          built_in_decls functions
  in

  let function_decl s = try StringMap.find s function_decls
       with Not_found -> raise (Failure ("unrecognized function " ^ s))
  in

  let _ = function_decl "main" in (* Ensure "main" is defined *)

  let check_function func =

    List.iter (check_not_void (fun n -> "illegal void formal " ^ n ^
      " in " ^ func.fname)) func.formals;

    report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ func.fname)
      (List.map snd func.formals);

    (* Type of each variable (global, formal, or local
       symbols is a reference, in order to be able to update the StringMap
       as you iterate through each statement in the function, to check
       that variables are declared before they are used
     *)
    let symbols = ref(List.fold_left (fun m (t, n) -> StringMap.add n t m)
       StringMap.empty (globals @ func.formals ))
    in

    let type_of_identifier s =
      try StringMap.find s !symbols
```

```
      with Not_found -> raise (Failure ("undeclared identifier " ^ s))
    in

    (* Return the type of an expression or throw an exception *)
    let rec expr = function
       IntLit _ -> Ptyp(Int)
      | DoubleLit _ -> Ptyp(Double)
      | StringLit _ -> Ptyp(String)
      | CharLit _   -> Ptyp(Char)
      | BoolLit _ -> Ptyp(Bool)
      | Id s -> type_of_identifier s
      | ListC el -> (match el with
                       [] -> Ptyp(Void)
                     | [e] -> expr e
                     | hd :: tl -> let et = expr hd
                                   in List.fold_left (fun a b -> if a = (expr b)
then a else raise (Failure("list contains different types of entries"))) et tl)
      | ArrayC el -> (match el with
                       [] -> Ptyp(Void)
                     | [e] -> expr e
                     | hd :: tl -> let et = expr hd
                                   in List.fold_left (fun a b -> if a = (expr b)
then a else raise (Failure("array contains different types of entries"))) et tl)
      | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
       (match op with
          Add | Sub | Mult | Div when t1 = Ptyp(Int) && t2 = Ptyp(Int) -> Ptyp(Int)
           | Add | Sub | Mult | Div when t1 = Ptyp(Double) && t2 = Ptyp(Double) ->
Ptyp(Double)
    | Equal | Neq when t1 = t2 -> Ptyp(Bool)
       | Less | Leq | Greater | Geq when t1 = Ptyp(Int) && t2 = Ptyp(Int) ->
Ptyp(Bool)
    | Less | Leq | Greater | Geq when t1 = Ptyp(Double) && t2 = Ptyp(Double) ->
Ptyp(Bool)
       | And | Or when t1 = Ptyp(Bool) && t2 = Ptyp(Bool) -> Ptyp(Bool)
        | _ -> raise (Failure ("illegal binary operator " ^
              string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
              string_of_typ t2 ^ " in " ^ string_of_expr e
              ^ "."
              ))
        )
      | Unop(op, e) as ex -> let t = expr e in
        (match op with
          Neg when t = Ptyp(Int) -> Ptyp(Int)
        | Not when t = Ptyp(Bool) -> Ptyp(Bool)
         | _ -> raise (Failure ("illegal unary operator " ^ string_of_uop op ^
                       string_of_typ t ^ " in " ^ string_of_expr ex)))
```

```
    | Noexpr -> Ptyp(Void)
    | Assign(var, e) as ex -> let lt = type_of_identifier var
                              and rt = expr e in
      check_assign (type_of_identifier var) (expr e)
          (Failure ("illegal assignment " ^ string_of_typ lt ^
          " = " ^ string_of_typ rt ^ " in " ^ string_of_expr ex))
    | Call("print", actuals) as call ->
      if List.length actuals == 1 then
          let et = expr (List.hd actuals) in
          let fd = function_decl ("print:" ^ string_of_typ et) in
          fd.typ
      else
          raise (Failure ("expecting 1 arguments in " ^
              string_of_expr call))

    | Call(fname, actuals) as call -> let fd = function_decl fname in
      if List.length actuals != List.length fd.formals then
          raise (Failure ("expecting " ^ string_of_int
          (List.length fd.formals) ^ " arguments in " ^ string_of_expr call))
      else
          List.iter2 (fun (ft, _) e -> let et = expr e in
              ignore (check_assign ft et
              (Failure ("illegal actual argument found " ^
              string_of_typ et ^ " expected " ^ string_of_typ ft ^
              " in " ^ string_of_expr e))))
            fd.formals actuals;
          fd.typ

    (* check actor instantiation with constructor signature of that actor *)
    | NewActor(aname, actuals) as ex -> let ad = actor_decl aname in
      if List.length actuals != List.length ad.aformals then
        raise (Failure ("expecting " ^ string_of_int (List.length ad.aformals)
        ^ " arguments in " ^ string_of_expr ex))
      else
          List.iter2 (fun (ft, _) e -> let et = expr e in
              ignore (check_assign ft et
              (Failure ("illegal actual argument found " ^
              string_of_typ et ^ " expected " ^ string_of_typ ft ^
              " in " ^ string_of_expr e))))
            ad.aformals actuals;
          Ptyp(Actor)
    | Send (_, _, _, _) ->
       Ptyp(Void)

  in
```

```
    let check_bool_expr e = if expr e <> Ptyp(Bool)
     then raise (Failure ("expected Boolean expression in " ^ string_of_expr e))
     else () in

    (* Verify a statement or throw an exception *)
    let rec stmt = function
       Block sl -> let rec check_block = function
           [Return _ as s] -> stmt s
         | Return _ :: _ -> raise (Failure "nothing may follow a return")
         | Block sl :: ss -> check_block (sl @ ss)
         | s :: ss -> stmt s ; check_block ss
         | [] -> ()
        in check_block sl
      | Expr e -> ignore (expr e)
      (* Vdecl updates symbols via reference to add a variable declaration *)
      | Vdecl (t, n) ->
        if (StringMap.mem n !symbols) then raise (Failure ("local variable " ^ n ^ "
already exists"));
        symbols := StringMap.add n t !symbols;
        check_not_void (fun n -> "illegal void local: " ^ n) (t, n);
        ignore((t, n))
      | Vdef vdef ->
        if (StringMap.mem vdef.vname !symbols) then raise (Failure ("local variable "
^ vdef.vname ^ " already exists"));
         symbols := StringMap.add vdef.vname vdef.vtyp !symbols;
        check_not_void (fun n -> "illegal void local: " ^ n) (vdef.vtyp, vdef.vname);
         ignore(vdef)
      (*      | Sdef sdef -> ignore(find_struct_decl sdef.styp);*)
      | Return e -> let t = expr e in if t = func.typ then () else
         raise (Failure ("return gives " ^ string_of_typ t ^ " expected " ^
                         string_of_typ func.typ ^ " in " ^ string_of_expr e))

      | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
      | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr e2;
                               ignore (expr e3); stmt st
      | While(p, s) -> check_bool_expr p; stmt s
      | _ -> ()
    in

    stmt (Block func.body)

  in
  List.iter check_function functions
```

codegen.ml

```
(* Code generation: translate takes a semantically checked AST and
produces LLVM IR
LLVM tutorial: Make sure to read the OCaml version of the tutorial
http://llvm.org/docs/tutorial/index.html
Detailed documentation on the OCaml LLVM library:
http://llvm.moe/
http://llvm.moe/ocaml/

Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*)
module L = Llvm
module A = Ast

module StringMap = Map.Make(String)

let translate (globals, functions, actors) =
  let context = L.global_context () in
  let the_module = L.create_module context "Theatr"
  and i32_t  = L.i32_type  context
  and i8_t   = L.i8_type   context
  and i1_t   = L.i1_type   context
  and void_t = L.void_type context
  and ptr_t  = L.pointer_type (L.i8_type context)
  and d_t    = L.double_type context in

  (* list of tids of actors that have been instantiated. This is a reference -
     not the functional way of doing it - but was easiest way to keep track of
     actors that have been created *)
  let max_actors = 1024 in

  (* create message struct of
     { int switchCase; void *functionArgumentStruct ; void *sender }     *)
  let msg_struct_type = L.named_struct_type context "msg_struct" in
  let _ = L.struct_set_body msg_struct_type
                            [|i32_t; L.pointer_type i8_t ; L.pointer_type i8_t|]
                            false
  in

  (* actor_address_struct contains info if actor is alive, the thread's
```

```
     tid (for joining), and its msgQueue
     { int alive ; int tid ; void *msgQueue }              *)
  let actor_address_struct_type  = L.named_struct_type context
"actor_address_struct" in
  let _ = L.struct_set_body actor_address_struct_type [| i32_t ; i32_t ;
L.pointer_type i8_t |] false in
  let arr = Array.make max_actors (L.const_int i32_t 0) in
  let init = L.const_array actor_address_struct_type arr in
  let global_actors = L.define_global "global_actors" init the_module in

  (* global to keep track of how many actors have been created so far
     start the index at 1 because the main function gets the 0th position *)
  let init = L.const_int i32_t 1 in
  let actor_count = L.define_global "actor_count" init the_module in

  let ltype_of_ptyp = function
      A.Int -> i32_t
    | A.Double -> d_t
    | A.Bool -> i1_t
    | A.Void -> void_t
    | A.Actor -> L.pointer_type actor_address_struct_type
    | A.String -> ptr_t
    | A.Char    -> i8_t
  in

  let ltype_of_ctyp = function
      _ -> i32_t in

  let ltype_of_typ = function
      A.Ptyp p -> ltype_of_ptyp p
    | A.Ctyp (c, p) -> ltype_of_ctyp (c, p) in

  let handle_addition typ =
    match typ with
    | "double"    -> L.build_fadd
    | "i32"       -> L.build_add
    | _           -> raise (Failure("incompatible type passed to addition operation:
" ^typ))
  in

  let handle_subtraction typ =
    match typ with
    | "double"    -> L.build_fsub
    | "i32"       -> L.build_sub
    | _           -> raise (Failure("incompatible type passed to subtraction
operation: " ^typ))
```

```
  in

  let handle_mult typ =
    match typ with
    | "double"    -> L.build_fmul
    | "i32"       -> L.build_mul
    | _           -> raise (Failure("incompatible type passed to multiplication
operation: " ^typ))
  in

  let handle_div typ =
    match typ with
    | "double"    -> L.build_fdiv
    | "i32"       -> L.build_sdiv
    | _           -> raise (Failure("incompatible type passed to division operation:
" ^typ))
  in

  let handle_arith_binop op typ1 typ2 =
    match typ1 with
    | typ2  -> (
      match op with
      | A.Add   -> handle_addition typ1
      | A.Sub   -> handle_subtraction typ1
      | A.Mult  -> handle_mult typ1
      | A.Div   -> handle_div typ1
      | _       -> raise (Failure("incompatible type passed to arithmetic operation:
" ^typ1))
    )
                 (*    | _     -> raise (Failure("incompatible types passed to
arithmetic operation: " ^typ1 ^typ2))    *)
  in

  let handle_equal typ =
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.Oeq
    | "i32"       -> L.build_icmp L.Icmp.Eq
    | _  -> raise (Failure("incompatible type passed to equal operation: " ^typ))
  in
  let handle_neq typ =
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.One
    | "i32"    -> L.build_icmp L.Icmp.Ne
    | _  -> raise (Failure("incompatible type passed to neg operation: " ^typ))
  in
  let handle_less typ =
```

```
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.Olt
    | "i32"    -> L.build_icmp L.Icmp.Slt
    | _  -> raise (Failure("incompatible type passed to less than operation: "
^typ))
  in
  let handle_leq typ =
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.Ole
    | "i32"    -> L.build_icmp L.Icmp.Sle
    | _  -> raise (Failure("incompatible type passed to less than or equal
operation: " ^typ))
  in
  let handle_greater typ =
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.Ogt
    | "i32"    -> L.build_icmp L.Icmp.Sgt
    | _  -> raise (Failure("incompatible type passed to greater than operation: "
^typ))
  in
  let handle_geq typ =
    match typ with
    | "double"    -> L.build_fcmp L.Fcmp.Oge
    | "i32"    -> L.build_icmp L.Icmp.Sge
    | _  -> raise (Failure("incompatible type passed to greater than or equal
operation: " ^typ))
  in
  let handle_comp_binop op typ1 typ2 =
    match typ1 with
    | typ2 -> (
      match op with
      | A.Equal     -> handle_equal typ1
      | A.Neq       -> handle_neq typ1
      | A.Less      -> handle_less typ1
      | A.Leq       -> handle_leq typ1
      | A.Greater   -> handle_greater typ1
      | A.Geq       -> handle_geq typ1
      | _           -> raise(Failure("invalid op"))
    )
                (*    | _             -> raise (Failure("incompatible types passed
to comparison operation: " ^typ1 ^typ2))*)
  in

      (* let list_arg_types = List.map (fun (t,_) -> ltype_of_typ t)
adecl.A.aformals in *)
      (* let type_array = Array.of_list list_arg_types in *)
```

```
      (* let name = adecl.A.aname in *)
      (* let struct_type = L.named_struct_type context (name ^ "_struct") in *)
      (* let _ = L.struct_set_body struct_type type_array false in *)
      (* StringMap.add name struct_type m in *)
(*
  let struct_decls =
    let struct_t m sd =
      let list_arg_types = List.map (fun (t,_)-> ltype_of_typ t) sd.A.elements in
      let type_array = Array.of_list list_arg_types in
      let name = sd.A.name in
      let struct_type = L.named_struct_type context ("struct_" ^ name) in
      let _ = L.struct_set_body struct_type type_array false in
      StringMap.add sd.A.name struct_type m in
    List.fold_left struct_t StringMap.empty structs in
 *)
  (* define struct types for queue related functions *)
  let struct_message_t = L.named_struct_type context "struct.message" in
  let _ = L.struct_set_body struct_message_t [| L.i32_type context ; L.pointer_type
i8_t ; L.pointer_type i8_t |] false in

  let struct_queue_t = L.named_struct_type context "struct.queue" in
  ignore(L.struct_set_body struct_queue_t [| L.pointer_type struct_queue_t ;
struct_message_t |] false);

  let struct_head_t = L.named_struct_type context "struct.head" in
  let _ =
    let struct_anon_t = L.named_struct_type context "struct.anon" in
    ignore(L.struct_set_body struct_anon_t [| L.i32_type context ; L.i32_type
context ; L.i64_type context ; L.i64_type context ; L.i64_type context ;
L.pointer_type i8_t ; L.i32_type context ; L.i32_type context |] false);

    let union_pthread_cond_t = L.named_struct_type context "union.pthread_cond_t" in
    ignore(L.struct_set_body union_pthread_cond_t [| struct_anon_t ; L.array_type
(L.pointer_type i8_t) 4 |] false);

    let struct__pthread_internal_slist = L.named_struct_type context
"struct.__pthread_internal_slist" in
    ignore(L.struct_set_body struct__pthread_internal_slist [| L.pointer_type
struct__pthread_internal_slist |] false);

    let union_anon_t = L.named_struct_type context "union.anon" in
    ignore(L.struct_set_body union_anon_t [| struct__pthread_internal_slist |]
false);

    let struct___pthread_mutex_s_t = L.named_struct_type context
"struct.__pthread_mutex_s"  in
```

```
    ignore(L.struct_set_body struct___pthread_mutex_s_t [| L.i32_type context ;
L.i32_type context ; L.i32_type context ; L.i32_type context ; L.i32_type context ;
union_anon_t |] false);

    let union_pthread_mutex_t_t = L.named_struct_type context
"union.pthread_mutex_t" in
    ignore(L.struct_set_body union_pthread_mutex_t_t [| struct___pthread_mutex_s_t
|] false);

    ignore(L.struct_set_body struct_head_t [| L.pointer_type struct_queue_t ;
union_pthread_cond_t ; union_pthread_mutex_t_t |] false);
  in

  let initialize_queue_t = L.function_type (L.pointer_type struct_head_t) [||] in
  let initialize_queue_func = L.declare_function "initialize_queue"
initialize_queue_t the_module in

  let enqueue_t = L.function_type (L.void_type context) [| L.pointer_type
struct_head_t ;
                                                           struct_message_t |] in
  let enqueue_func = L.declare_function "enqueue" enqueue_t the_module in

  let dequeue_t = L.function_type (L.void_type context) [| L.pointer_type
struct_message_t ; L.pointer_type struct_head_t |] in
  let dequeue_func = L.declare_function "dequeue" dequeue_t the_module in

  (* declare geturl function *)
  let geturl_func_t = L.function_type (L.i32_type context) [| L.pointer_type i8_t ;
L.pointer_type i8_t |] in
  let geturl_func = L.declare_function "geturl" geturl_func_t the_module in

  (********* END of define variable for queu related function **************)

  (* Declare each global variable; remember its value in a map *)
  let global_vars =
    let global_var m (t, n) =
      let init = L.const_int (ltype_of_typ t) 0
      in StringMap.add n (L.define_global n init the_module) m in
    List.fold_left global_var StringMap.empty globals in

  (* Declare printf(), which the print built-in function will call *)
  let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
  let printf_func = L.declare_function "printf" printf_t the_module in

  (* Declare pthread_create() and pthread_join(), which will be called to spawn
actors *)
```

```
  let pthread_t = i32_t
  and func_void_ptr_void_ptr = L.function_type (L.pointer_type i8_t) [|
L.pointer_type i8_t |]
  in

  let pthread_create_t_arr = [|L.pointer_type i32_t ; L.pointer_type i8_t;
    L.pointer_type func_void_ptr_void_ptr ; L.pointer_type i8_t |]
  in
  let pthread_create_t = L.function_type i32_t pthread_create_t_arr in
  let pthread_create_func = L.declare_function "pthread_create" pthread_create_t
the_module in

  let pthread_join_arr = [| pthread_t ; L.pointer_type (L.pointer_type i8_t) |] in
  let pthread_join_t = L.function_type i32_t pthread_join_arr in
  let pthread_join_func = L.declare_function "pthread_join" pthread_join_t
the_module in



  (* Define each function (arguments and return type) so we can call it *)
  let function_decls =
    let function_decl m fdecl =
      let name = fdecl.A.fname
      and formal_types =
  Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.A.formals)
      in let ftype = L.function_type (ltype_of_typ fdecl.A.typ) formal_types in
      StringMap.add name (L.define_function name ftype the_module, fdecl) m in
    List.fold_left function_decl StringMap.empty functions in


  (* construct the struct types that will hold the arguments for each actor
     a pointer to an instance of these structs will be passed in during the pthread
call *)
  let actor_struct_types =
    let actor_struct m adecl =
      let list_arg_types = List.map (fun (t,_) -> ltype_of_typ t) adecl.A.aformals
in
      (* prepend the invisible argument of the index number of the actor *)
      let list_arg_types = i32_t :: list_arg_types in
      let type_array = Array.of_list list_arg_types in
      let name = adecl.A.aname in
      let struct_type = L.named_struct_type context (name ^ "_struct") in
      let _ = L.struct_set_body struct_type type_array false in
      StringMap.add name struct_type m in
    List.fold_left actor_struct StringMap.empty actors in

  let (msg_functions, _) =
```

```
    let msg_function (m, count) adecl =
      let msg_function_recv (m, count) mdecl =
        let namespaced_id = adecl.A.aname ^ "." ^ mdecl.A.mname in
        StringMap.add namespaced_id count m, count+1
      in
      List.fold_left msg_function_recv (m, count) adecl.A.receives
    in
    List.fold_left msg_function (StringMap.empty, 1) actors
  in
  let msgLookup n = try StringMap.find n msg_functions
                    with Not_found -> -1
  in

  (* construct the actor's main looping function to pass to pthread_create
     This will be almost the same for each actor, the difference being the
     specific local variables that each actor needs to maintain its state
     on its thread's stack *)
  let actor_decls =
    let actor_decl m adecl =
      let name = adecl.A.aname in
      let atype = L.function_type (L.pointer_type i8_t) [|L.pointer_type i8_t|] in
      (* type of the function is void function(void star), for pthread *)
      let count_mdecl (count, m) decl =
        let m = StringMap.add decl.A.mname count m in
        (count+1, m)
      in
      let (_, funcMap) = List.fold_left count_mdecl (1, StringMap.empty)
adecl.A.receives in
      StringMap.add name (L.define_function name atype the_module, adecl, funcMap) m
in
    List.fold_left actor_decl StringMap.empty actors in

  let local_actors = ref StringMap.empty in
  let local_vars = ref StringMap.empty in
  let lookup n = try StringMap.find n !local_vars
                 with Not_found -> try StringMap.find n global_vars
                                   with Not_found -> raise (Failure ("undeclared
variable " ^ n))
  in

  let rec expr builder = function
      A.IntLit i -> L.const_int i32_t i
    | A.DoubleLit f -> L.const_float d_t f
    | A.StringLit s ->
       let format_str_str s = L.build_global_stringptr (s) ".str" builder in
       format_str_str s
```

```
    | A.CharLit c -> L.const_int i8_t (Char.code c)
    | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)
    | A.Noexpr -> L.const_int i32_t 0
    | A.Id s -> L.build_load (lookup s) s builder
    | A.Binop (e1, op, e2) ->
       let e1' = expr builder e1
       and e2' = expr builder e2 in

       let typ_e1 = L.string_of_lltype(L.type_of e1')
       and typ_e2 = L.string_of_lltype(L.type_of e2') in
       (match op with
        | A.Add | A.Sub | A.Mult | A.Div   -> handle_arith_binop op typ_e1 typ_e2
        | A.And     -> L.build_and
        | A.Or      -> L.build_or
        | A.Equal | A.Neq | A.Less | A.Leq | A.Greater | A.Geq  -> handle_comp_binop
op typ_e1 typ_e2
       ) e1' e2' "tmp" builder
    | A.Unop(op, e) ->
       let e' = expr builder e in
       (match op with
          A.Neg     -> L.build_neg
        | A.Not     -> L.build_not) e' "tmp" builder
    | A.Assign (s, e) -> let e' = expr builder e in
                ignore (L.build_store e' (lookup s) builder); e'
    | A.Call ("geturl", [e1; e2]) ->
       L.build_call geturl_func [| (expr builder e1) ; (expr builder e2) |] "geturl"
builder

    | A.Call ("print", [e]) | A.Call ("printb", [e]) ->
       let format_int_str = L.build_global_stringptr "%d\n" "fmt" builder
       and format_double_str = L.build_global_stringptr "%f\n" "fmt" builder in
       let format_str_str = L.build_global_stringptr ("%s\n") "fmt" builder in
       let format_char_str = L.build_global_stringptr "%c\n" "fmt" builder in

       let get_format_typ_str typ =
         match typ with
         | "i32" -> format_int_str
         | "double" -> format_double_str
         | "i8"   -> format_char_str
         (* | "char" -> format_char_str *)
         | _    -> raise (Failure("invalid type passed to print, "^typ))
       in
       let e1 = expr builder e in
       let typ_e = L.string_of_lltype (L.type_of e1) in
       if typ_e = "i8*" then
         L.build_call printf_func [| format_str_str ; e1 |] "printf" builder
```

```
       else if typ_e = "i1" then
         L.build_call printf_func [| format_int_str ; (expr builder e) |] "printf"
builder
       else
         let format_typ_str = get_format_typ_str typ_e in
         L.build_call printf_func [| format_typ_str; e1 |] "printf" builder
    | A.Call (f, act) ->
       let (fdef, fdecl) = StringMap.find f function_decls in
       let actuals = List.rev (List.map (expr builder) (List.rev act)) in
       let result = (match fdecl.A.typ with A.Ptyp(A.Void) -> ""
                                          | _ -> f ^ "_result") in
       L.build_call fdef (Array.of_list actuals) result builder
    (* codegen for actors: create a new struct on the stack to store arguments
         and pass a pointer to the struct, along with a pointer to the actor's
         function to pthread *)
    | A.NewActor (a, act) ->
       let (adef, _, _) = StringMap.find a actor_decls in
       let a_struct_type = StringMap.find a actor_struct_types in
       let actuals = List.rev (List.map (expr builder) (List.rev act)) in

       (* get position in global actors array *)
       let curr_actor_count = L.build_load actor_count "curr_actor_count" builder in

       let zero = L.const_int i32_t 0 in
       let addr_struct = L.build_in_bounds_gep global_actors
                         [| zero ; curr_actor_count |] "pos" builder in
       let alive = L.build_struct_gep addr_struct 0 "" builder in
       let _ = L.build_store (L.const_int i32_t 1) alive builder in
       let tid_in_struct = L.build_struct_gep addr_struct 1 "" builder in
       let msgQueue = L.build_struct_gep addr_struct 2 "" builder in
       let qhead = L.build_call initialize_queue_func [||] "" builder in
       let qhead_casted = L.build_bitcast qhead (L.pointer_type i8_t) "" builder in
       let _ = L.build_store qhead_casted msgQueue builder in
       let result = addr_struct in

       (* create the actor's function argument struct on the heap
          this will be freed by the actor's pthread function after the args are
          copied onto that thread's stack *)
       let a_struct = L.build_malloc a_struct_type "" builder in
       (* fill in the struct with actuals, prepend the addr_struct as an invisible
argument *)
       let actuals = curr_actor_count :: actuals in
       let index_and_store idx actual =
         let ptr = L.build_struct_gep a_struct idx "" builder in
         let _ = L.build_store actual ptr builder in
         idx+1 in
```

```
      let _ = (match List.length actuals with
                 0 -> -1
               | _ -> List.fold_left index_and_store 0 actuals)
      in
      let a_struct_ptr_casted = (match List.length actuals with
                                   0 -> L.const_bitcast (L.const_pointer_null i8_t)
(L.pointer_type i8_t)
                                 | _ -> L.build_bitcast a_struct (L.pointer_type
i8_t) "" builder) in
      let pthread_pt = L.build_alloca i32_t "tid" builder in
      let attr = L.const_bitcast (L.const_pointer_null i8_t) (L.pointer_type i8_t)
in
      let pthread_args = [| pthread_pt ; attr ; adef ; a_struct_ptr_casted |] in
      let _ = L.build_call pthread_create_func pthread_args "pthread_create_result"
builder in
      (* store the value of the tid into the actor_address_struct that you made
before *)
      let tid_val = L.build_load pthread_pt "" builder in
      let _ = L.build_store tid_val tid_in_struct builder in
      (* increment actor count *)
      let new_actor_count = L.build_add curr_actor_count (L.const_int i32_t 1) ""
builder in
      let _ = L.build_store new_actor_count actor_count builder in
      result
    | A.Send (actorType, msgFunction, msgArgs, recipientName) ->
      let llvalue = lookup recipientName in
      let addr_struct = L.build_load llvalue "addr_struct" builder in

      let msgQueue_ptr_idx = L.build_struct_gep addr_struct 2 "msgQueue_ptr"
builder in
      let msgQueue_raw = L.build_load msgQueue_ptr_idx "msgQueue" builder in
      let msgQueue = L.build_bitcast msgQueue_raw (L.pointer_type struct_head_t) ""
builder in
      let message = L.build_alloca struct_message_t "" builder in
      let actuals = List.rev (List.map (expr builder) (List.rev msgArgs)) in
      let actuals_array = Array.of_list actuals in
      let type_ptr_list = List.map L.type_of actuals in
      let type_ptr_array = Array.of_list type_ptr_list in
      let struct_type = L.named_struct_type context "temp" in
      let _ = L.struct_set_body struct_type type_ptr_array false in

      let argument_struct = L.build_malloc struct_type "argument_struct" builder in

      let index_and_store idx _ =
        let ptr = L.build_alloca ((type_ptr_array).(idx)) "" builder in
        let _ = L.build_store actuals_array.(idx) ptr builder in
```

```
        let arg_val = L.build_load ptr "arg_val" builder in
        let argument_ptr = L.build_struct_gep argument_struct idx "" builder in
        let _ = L.build_store arg_val argument_ptr builder in idx + 1 in
      let _ = (match List.length actuals with
                 0 -> -1
               | _ -> List.fold_left index_and_store 0 type_ptr_list)
      in
      let argument_struct_casted = (match List.length actuals with
                 0 -> L.build_bitcast (L.const_pointer_null i8_t) (L.pointer_type
i8_t) "" builder
               | _ -> L.build_bitcast argument_struct (L.pointer_type i8_t) ""
builder)
      in
      let sender_ptr = L.const_bitcast (L.const_pointer_null i8_t) (L.pointer_type
i8_t) in

      (* fill out the msgqueue struct with the case and the pointers to
funcArgsStruct and sender ptrs *)
      let namespaced_msgFunc = actorType ^ "." ^ msgFunction in
      let case_num = msgLookup namespaced_msgFunc in
      let case_num = match msgFunction with
        | "die" -> 0
        | _ -> case_num
      in
      let case_val = (L.const_int i32_t case_num) in
      let case_ptr = L.build_alloca i32_t "case" builder in
      let _ = L.build_store case_val case_ptr builder in
      let msg_case_ptr = L.build_struct_gep message 0 "" builder in
      let msg_arg_struct_ptr = L.build_struct_gep message 1 "" builder in
      let msg_sender_ptr = L.build_struct_gep message 2 "" builder in
      let _ = L.build_store case_val msg_case_ptr builder in
      let _ = L.build_store argument_struct_casted msg_arg_struct_ptr builder in
      let _ = L.build_store sender_ptr msg_sender_ptr builder in
      let message_val = L.build_load message "message_val" builder in
      let _ = L.build_call enqueue_func [| msgQueue ; message_val |] "" builder in
      L.const_int i32_t 1
    | _ -> raise(Failure("unrecognized expression"))
  in


  (* Fill in the body of the each actor's thread function *)
  let build_actor_thread_func_body adecl =
    let (the_function, _, _) = StringMap.find adecl.A.aname actor_decls in
    let builder = L.builder_at_end context (L.entry_block the_function) in
    (* Construct the thread function's "locals":
       Since this function is passed into pthread, it will have to
```

```
       create a struct pointer that matches its arguments and dereference
       the passed in ptr argument to fill out the argument variables on the stack
       ex:
       void *dolphin_actor(void *ptr) {
           struct dolphin_state *s = ptr;
           int height = s->height;
           int weight = s->weight;
       }
    *)
    (* create the struct pointer - steps copied from LLVM emitted by C code *)
    let voidp = Array.get (L.params the_function) 0 in
    let _ = L.set_value_name "ptr" voidp in
    let local_voidp = L.build_alloca (L.pointer_type i8_t) "" builder in
    let _ =  L.build_store voidp local_voidp builder in
    let struct_type = StringMap.find adecl.A.aname actor_struct_types in
    let local_struct_p = L.build_alloca (L.pointer_type struct_type) "state_struct"
builder in
    let loaded_voidp = L.build_load local_voidp "" builder in
    let casted_voidp = L.build_bitcast loaded_voidp (L.pointer_type struct_type) ""
builder in
    let _ = L.build_store casted_voidp local_struct_p builder in
    local_vars := StringMap.empty;
    local_actors := StringMap.empty;

    (* add the "fake" first argument - the pointer to the addr_struct to local_vars
*)
    let load_struct = L.build_load local_struct_p "" builder in
    let var_at_idx = L.build_struct_gep load_struct 0 "" builder in
    let var_stored = L.build_load var_at_idx "" builder in
    let local = L.build_alloca i32_t "self_index" builder in
    ignore (L.build_store var_stored local builder);
    local_vars := StringMap.add "self:index" local !local_vars;

    (* create the formals as local variables, add them to locals map *)
    let add_formal (idx, struct_p) (t, n) =
      let load_struct = L.build_load struct_p "" builder in
      (* need to cast 0 and idx to be i32_t before passing to index array in LLVM *)
      let zero = L.const_int i32_t 0 in
      let idxVal = L.const_int i32_t idx in
      let var_at_idx = L.build_in_bounds_gep load_struct [|zero; idxVal|] "" builder
in
      let var_stored = L.build_load var_at_idx "" builder in
      let local = L.build_alloca (ltype_of_typ t) n builder in
      ignore (L.build_store var_stored local builder);
      local_actors := StringMap.add n (local, StringMap.empty) !local_actors;
      local_vars := StringMap.add n local !local_vars; (idx+1, struct_p) in
```

```
    let add_local stmt = match stmt with
      | A.Vdecl (t, n) ->
         let local_var = L.build_alloca (ltype_of_typ t) n builder in
         local_actors := StringMap.add n (local_var, StringMap.empty) !local_actors;
         local_vars := StringMap.add n local_var !local_vars
      | A.Vdef v ->
         let local_var = L.build_alloca (ltype_of_typ v.A.vtyp) v.A.vname builder in
         local_actors := StringMap.add v.A.vname (local_var, StringMap.empty)
!local_actors;
         local_vars := StringMap.add v.A.vname local_var !local_vars
      | _ -> ()
    in
    let _ = List.fold_left add_formal (1, local_struct_p) adecl.A.aformals in
    List.iter add_local adecl.A.alocals;
    ignore(L.build_free voidp builder); (* free the malloc'd function arguments
struct *)

    (* Invoke "f builder" if the current block doesn't already
       have a terminal (e.g., a branch). *)
    let add_terminal builder f =
      match L.block_terminator (L.insertion_block builder) with
        Some _ -> ()
      | None -> ignore (f builder) in

    let rec stmt builder = function
        A.Block sl -> List.fold_left stmt builder sl
      | A.Expr e -> ignore (expr builder e); builder
      | A.Vdecl v -> ignore (v); builder (* we've already added this to locals *)
      | A.Vdef v -> ignore (expr builder v.A.vvalue); builder
      | A.Sdef s -> ignore (s); builder
      | A.If (predicate, then_stmt, else_stmt) ->
         let bool_val = expr builder predicate in
         let merge_bb = L.append_block context "merge" the_function in

         let then_bb = L.append_block context "then" the_function in
         add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
                      (L.build_br merge_bb);

         let else_bb = L.append_block context "else" the_function in
         add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
                      (L.build_br merge_bb);

         ignore (L.build_cond_br bool_val then_bb else_bb builder);
         L.builder_at_end context merge_bb
```

```
      | A.While (predicate, 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
         add_terminal (stmt (L.builder_at_end context body_bb) body)
                      (L.build_br pred_bb);

         let pred_builder = L.builder_at_end context pred_bb in
         let bool_val = expr pred_builder predicate in

         let merge_bb = L.append_block context "merge" the_function in
         ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
         L.builder_at_end context merge_bb

      | A.For (e1, e2, e3, body) -> stmt builder
      ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
      | _ -> raise(Failure("invalid stament in actor"))
    in

    (* Execute actor's local statements *)
    let builder = L.builder_at_end context (L.entry_block the_function) in
    let builder = stmt builder (A.Block adecl.A.alocals) in

    (***** Actor implementation
     * Constant communication with message queue simulated through infite while
loop.
     * Different message functionality simulated through switch-cases
     * die() message exits infite while loop.
     *
     * ******)
    let make_block name =
        let bb = L.append_block context (name^"_bb") the_function in
        let builder = L.builder_at_end context bb in
        (bb, builder)
    in

    let build_pred_block pred bb builder body_bb merge_bb =
        let bool_val = pred in

        (* Terminator for pred block *)
        L.position_at_end bb builder;
        ignore (L.build_cond_br bool_val body_bb merge_bb builder);
        builder
    in
```

```
    (* Returns list of msg blocks *)
    let build_msg_blocks finish_bb merge_bb =
        let create_msg_block m decl =
          let namespace_id = adecl.A.aname ^ "." ^ decl.A.mname in
          let (msg_bb, msg_builder) = make_block ("msg_"^decl.A.mname^"_case") in
          L.position_at_end msg_bb msg_builder;
          StringMap.add namespace_id msg_bb m in

        let create_default_msg_block decl =
            let actor_local_vars_copy = ! local_vars in
            let (msg_bb, msg_builder) = make_block "msg_default_case" in
            List.iter add_local decl.A.dabody;
            let msg_builder = stmt msg_builder (A.Block decl.A.dabody) in
            local_vars := actor_local_vars_copy;  (* Resets actor local vars *)
            ignore(L.build_br finish_bb msg_builder);
            msg_bb in

        let create_die_msg_block =
            let (die_bb, die_builder) = make_block "msg_die_case" in
            ignore(L.build_br merge_bb die_builder);
            die_bb in

        let cases = List.fold_left create_msg_block StringMap.empty adecl.A.receives
in
        let def_bb = create_default_msg_block adecl.A.drop in
        let die_bb = create_die_msg_block in
        let cases = StringMap.add "drop" def_bb cases in
        let cases = StringMap.add "die" die_bb cases in
        cases

    in

    (* Adds msg instructions - local vars and stmts *)
    let add_msg_instructions decl bb struct_typ actuals_ptr _ finish_bb =
        let actor_local_vars_copy = !local_vars in
        let actor_local_actors_copy = !local_actors in
        let msg_builder = L.builder_at_end context bb in

        let msg_struct_ptr_casted = L.build_bitcast actuals_ptr (L.pointer_type
struct_typ) "actual_ptr" msg_builder in
        let msg_struct_ptr_p = L.build_alloca (L.pointer_type struct_typ) ""
msg_builder in
        let _ = L.build_store msg_struct_ptr_casted msg_struct_ptr_p msg_builder in
        let msg_struct_ptr = L.build_load msg_struct_ptr_p "" msg_builder in
        let add_msg_formal idx (t,n) =
            let ptr = L.build_struct_gep msg_struct_ptr idx "f_ptr" msg_builder in
```

```
            let value = L.build_load ptr "" msg_builder in
            let local = L.build_alloca (ltype_of_typ t) n msg_builder in
            ignore(L.build_store value local msg_builder);
            local_vars := StringMap.add n local !local_vars;
            (idx+1) in
        let _ = List.fold_left add_msg_formal 0 decl.A.mformals in

        List.iter add_local decl.A.mbody;
        let msg_builder = stmt msg_builder (A.Block decl.A.mbody) in
        local_vars := actor_local_vars_copy;  (* Resets actor local vars *)
        local_actors := actor_local_actors_copy; (* Resets actor local actors *)
        ignore(L.build_br finish_bb msg_builder);
    in

    let build_body_block bb builder finish_bb merge_bb =
        (* Map of message structs - decl name : struct_type *)
        let msg_struct_types =
            let msg_struct m decl =
                let list_arg_types = List.map (fun (t,_) -> ltype_of_typ t)
decl.A.mformals in
                let type_array = Array.of_list list_arg_types in
                let name = decl.A.mname in
                let struct_type = L.named_struct_type context (name ^ "_struct") in
                let _ = L.struct_set_body struct_type type_array false in
                StringMap.add name struct_type m in
            List.fold_left msg_struct StringMap.empty adecl.A.receives in
        (* pull message off queue and store the args struct pointer on the stack,
           along with the message number*)

        let self_index = lookup "self:index" in
        let self_index_val = L.build_load self_index "self:index_val" builder in
        let zero = L.const_int i32_t 0 in
        let self_addr_struct = L.build_in_bounds_gep global_actors
                               [| zero ; self_index_val |] "pos" builder in

        let msgQueue_ptr_idx = L.build_struct_gep self_addr_struct 2 "" builder in
        let msgQueue_raw = L.build_load msgQueue_ptr_idx "" builder in
        let msgQueue = L.build_bitcast msgQueue_raw (L.pointer_type struct_head_t)
"" builder in
        let ret_message_struct = L.build_alloca struct_message_t "message_struct"
builder in
        let _ = L.build_call dequeue_func [| ret_message_struct ; msgQueue |] ""
builder in
        let loaded_local_msg_struct = ret_message_struct in
        let idx_case = L.build_struct_gep loaded_local_msg_struct 0 "" builder in
        let case = L.build_load idx_case "case_num" builder in (* i32 *)
```

```
        let idx_actuals_ptr = L.build_struct_gep loaded_local_msg_struct 1 ""
builder in (* *)
        let actuals_ptr = L.build_load idx_actuals_ptr "actuals_ptr" builder in
        let idx_sender_ptr = L.build_struct_gep loaded_local_msg_struct 2 "" builder
in
        let sender_ptr = L.build_load idx_sender_ptr "sender_ptr" builder in
        let cases = build_msg_blocks finish_bb merge_bb in

        (* Terminator for body block *)
        L.position_at_end bb builder;
        let sw = L.build_switch case (StringMap.find "drop" cases)
                                ((List.length adecl.A.receives)+1) builder in

        (*Adds cases to switch and creates block and decl maps*)
        let add_msg_to_switch name bb =
          if name = "drop" then () else
            let num = match name with
              | "die" -> 0
              | _ -> msgLookup name
            in
            let case_num = L.const_int i32_t num in
            let _ = L.add_case sw case_num bb in
            ()
        in
        let _ = StringMap.iter add_msg_to_switch cases in

        (* Adds msg body instructions msg body *)
        let add_msg_vars_body decl =
          let namespace_id = adecl.A.aname ^ "." ^ decl.A.mname in
          let msg_bb = StringMap.find namespace_id cases in
          let mstruct_t = StringMap.find decl.A.mname msg_struct_types in
          add_msg_instructions decl msg_bb mstruct_t actuals_ptr sender_ptr
finish_bb
        in
        List.iter add_msg_vars_body adecl.A.receives;
        builder
    in

    let build_merge_block bb builder =
        let ret_void_star = L.build_alloca (i8_t) "ret" builder in

        (* Terminator for merge block *)
        L.position_at_end bb builder;
        ignore(L.build_ret ret_void_star builder);
        builder
    in
```

```
    let build_actor_while =
        let (pred_bb, pred_builder) = make_block "pred_msg_while" in
        let (body_bb, body_builder) = make_block "body_msg_while" in
        let (finish_bb, finish_builder) = make_block "finish_msg_while" in
        let (merge_bb, merge_builder) = make_block "merge_msg_while" in

        ignore(L.build_br pred_bb finish_builder); (* Terminator for finish block *)

        (* Adds instructions and terminators for pred, body, and merge blocks *)
        let pred = L.const_int i1_t 1 in (* if 1, while loop runs *)
        let _ = build_pred_block pred  pred_bb pred_builder body_bb merge_bb in
        let _ = build_body_block body_bb body_builder finish_bb merge_bb in
        let _ = build_merge_block merge_bb merge_builder in

        ignore (L.build_br pred_bb builder); (* Terminator for block calling while
*)
        ()
    in
    build_actor_while in
  let _ = List.iter build_actor_thread_func_body actors in
  (* finished building actor thread functions, move on to normal functions *)


  (* Fill in the body of the given function *)
  let build_function_body fdecl =
    let (the_function, _) = StringMap.find fdecl.A.fname function_decls 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 *)
    local_vars := StringMap.empty;
    let add_formal (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);
    local_vars := StringMap.add n local !local_vars;
    local_actors := StringMap.add n (local, StringMap.empty) !local_actors in
    List.iter2 add_formal fdecl.A.formals (Array.to_list (L.params the_function));
    (* make a pass through the function body for variable declarations
         and add them to local_vars. This assumes that semantic checker
         has gone through to make sure variables have been declared before
         use *)
    let add_local stmt = match stmt with
      | A.Vdecl (t, n) ->
          let local_var = L.build_alloca (ltype_of_typ t) n builder
```

```
          in local_vars := StringMap.add n local_var !local_vars;
              local_actors := StringMap.add n (local_var, StringMap.empty)
!local_actors
        | A.Vdef v ->
          let local_var = L.build_alloca (ltype_of_typ v.A.vtyp) v.A.vname builder
          in local_vars := StringMap.add v.A.vname local_var !local_vars;
              local_actors := StringMap.add v.A.vname (local_var, StringMap.empty)
!local_actors
        | _ -> ()
    in
    List.iter add_local fdecl.A.body;

    (* Invoke "f builder" if the current block doesn't already
       have a terminal (e.g., a branch). *)
    let add_terminal builder f =
      match L.block_terminator (L.insertion_block builder) with
        Some _ -> ()
      | None -> ignore (f builder) in

    (* Build the code for the given statement; return the builder for
       the statement's successor *)
    let rec stmt builder = function
        A.Block sl -> List.fold_left stmt builder sl
      | A.Expr e -> ignore (expr builder e); builder
      | A.Vdecl v -> ignore (v); builder (* we've already added this to locals *)
      | A.Vdef v -> ignore (expr builder v.A.vvalue); builder
      | A.Return e ->
         (* if building the main() function, add pthread_joins before the return *)
         let builder =
           if fdecl.A.fname = "main" then
             (* build a for loop through global array, running pthread join on the
tids *)
             let i = L.build_alloca i32_t "" builder in
             local_vars := StringMap.add "return:i" i !local_vars;
             let e1 = A.Assign ("return:i", A.IntLit 1) in (* start index at 1 *)
             let e2 = A.Binop (A.Id "return:i", A.Less, A.IntLit max_actors) in
             let add_1_expr = A.Binop (A.Id "return:i", A.Add, A.IntLit 1) in
             let e3 = A.Assign ("return:i", add_1_expr) in

             let builder = stmt builder (A.Expr e1) in
             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 = L.builder_at_end context body_bb in
             let idx = L.build_load i "idx" builder in
```

```
            let zero = L.const_int i32_t 0 in
            let addr_struct = L.build_in_bounds_gep global_actors
                                 [| zero ; idx |] "pos" builder in
            let tid_p = L.build_struct_gep addr_struct 1 "tid_p" builder in
            let tid_p_cast = L.build_bitcast tid_p (L.pointer_type i32_t) ""
builder in
            let tid_val = L.build_load tid_p_cast "tid_val" builder in
            let join_attr = L.const_bitcast (L.const_pointer_null i8_t)
                           (L.pointer_type (L.pointer_type i8_t)) in
            ignore(L.build_call pthread_join_func [| tid_val ; join_attr |]
"pthread_join_result" builder);
            ignore(stmt builder (A.Expr e3));
            add_terminal builder (L.build_br pred_bb);
            let pred_builder = L.builder_at_end context pred_bb in
            let bool_val = expr pred_builder e2 in
            let merge_bb = L.append_block context "merge" the_function in
            ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
            L.builder_at_end context merge_bb
          else
            builder
        in
        ignore (match fdecl.A.typ with
              A.Ptyp(A.Void) -> L.build_ret_void builder
        | _ -> L.build_ret (expr builder e) builder); builder
      | A.Sdef s -> ignore (s); builder
      | A.If (predicate, then_stmt, else_stmt) ->
        let bool_val = expr builder predicate in
        let merge_bb = L.append_block context "merge" the_function in

        let then_bb = L.append_block context "then" the_function in
        add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
                     (L.build_br merge_bb);

        let else_bb = L.append_block context "else" the_function in
        add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
                     (L.build_br merge_bb);

        ignore (L.build_cond_br bool_val then_bb else_bb builder);
        L.builder_at_end context merge_bb

      | A.While (predicate, 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
        add_terminal (stmt (L.builder_at_end context body_bb) body)
```

```
                    (L.build_br pred_bb);

        let pred_builder = L.builder_at_end context pred_bb in
        let bool_val = expr pred_builder predicate in

        let merge_bb = L.append_block context "merge" the_function in
        ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
        L.builder_at_end context merge_bb

     | A.For (e1, e2, e3, body) -> stmt builder
     ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
   in

   (* Build the code for each statement in the function *)
   let builder = stmt builder (A.Block fdecl.A.body) in

   (* Add a return if the last block falls off the end *)
   add_terminal builder (match fdecl.A.typ with
                         | A.Ptyp(A.Void) -> L.build_ret_void
                         | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
  in
  List.iter build_function_body functions;
  the_module
```

Makefile

```
# Make sure ocamlbuild can find opam-managed packages: first run
#
# eval `opam config env`

# Easiest way to build: using ocamlbuild, which in turn uses ocamlfind
all : theatr.native queue.o filedwld.o

theatr.native :
      ocamlbuild -use-ocamlfind -pkgs llvm,llvm.analysis,llvm.bitreader,llvm.linker
-cflags -w,+a-4 \
             theatr.native

# "make clean" removes all generated files

.PHONY : clean
clean :
      ocamlbuild -clean
```

```
	rm -rf testall.log *.diff theatr scanner.ml parser.ml parser.mli
	rm -rf *.cmx *.cmi *.cmo *.cmx *.o
	rm -rf *.ll *.ll
	rm -rf *.ll *.temp *.out *.err *.s *.exe *.pdf


# More detailed: build using ocamlc/ocamlopt + ocamlfind to locate LLVM

OBJS = ast.cmx codegen.cmx parser.cmx scanner.cmx semant.cmx theatr.cmx
POBJS = scanner.cmo parser.cmo ast.cmo semant.cmo print_ast.cmo

theatr : $(OBJS)
	ocamlfind ocamlopt -linkpkg -package llvm -package llvm.analysis $(OBJS) -o
theatr

print_ast : $(POBJS)
	ocamlc -o print_ast $(POBJS)

clean_ast :
	rm -f print_ast parser.ml parser.mli scanner.ml *.cmo *.cmi

scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	ocamlyacc parser.mly

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

%.cmx : %.ml
	ocamlfind ocamlopt -c -package llvm $<

queue : queue.c
	cc -o queue -DBUILD_TEST queue.c

filedwld : filedwld.c
	cc -o filedwld -DBUILD_TEST filedwld.c

### Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
ast.cmo :
ast.cmx :
codegen.cmo : ast.cmo
```

```
codegen.cmx : ast.cmx
theatr.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
theatr.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmx parser.cmi
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx
semant.cmo : ast.cmo
semant.cmx : ast.cmx
parser.cmi : ast.cmo

# Building the tarball

TESTS = arith1 arith2 arith3 fib for1 for2 func1 func2 func3 func4 \
    func5 gcd2 gcd global1 global2 hello if1 if2 if3 if4 ops1 ops2 \
    var1 while1 local1

FAILS = assign1 assign2 assign3 dead1 dead2 expr1 expr2 for1 for2  \
    for3 for4 for5 func1 func2 func3 func4 func5 func6 func7 func8 \
    func9 global1 global2 if1 if2 if3 nomain return1 return2 while1      \
    while2

TESTFILES = $(TESTS:%=test-%.mc) $(TESTS:%=test-%.out) \
          $(FAILS:%=fail-%.mc) $(FAILS:%=fail-%.err)

TARFILES = ast.ml codegen.ml Makefile theatr.ml parser.mly README scanner.mll \
       semant.ml testall.sh $(TESTFILES:%=tests/%)

microc-llvm.tar.gz : $(TARFILES)
       cd .. && tar czf microc-llvm/microc-llvm.tar.gz \
             $(TARFILES:%=microc-llvm/%)
```

preprocessor.py

```
#!/usr/bin/python3
""" Preprocessor for Theatr compiler
Author: Suraj Keshri
"""
import sys
import re


def insert(s, char, p):
    if (p > len(s)+1 or p < 0):
        raise ValueError("p can't be larger that string length {}".format(len(s)))
    else:
        if p == 0:
            return char+s
        else:
            return s[:p]+char+s[p:]

def insert_mult_char(s, char_pos):
    for i, a in enumerate(char_pos):
        s = insert(s, a[0], a[1]+i)
    return s

def update_stack(stack, new_point, result):
    """
      Each entry in the stack is (a = new_line_character_position, b = number of
tabs following new line)
      Iterate through the stack in reverse order. The stack always
      has bigger blocks above smaller block. The order is defined by b.

      INPUT: stack, new stack entry, a list where result would be appended
      result is updated with tuples ('{' or '}', position where brace should be
inserted)
    """
    stack_rev = reversed(stack)
    insert_lbrace_disabled = 0
    for a in stack_rev:
        # if new block is smaller or of equal size, keep deleting the blocks below
        if a[1] > new_point[1]:
            # print('insert rbrace at {}'.format(new_point[0]))
            result.append(('}', new_point[0]))
            # can't insert a lbrace once a rbrace is inserted
            insert_lbrace_disabled = 1
            del stack[-1]
        if a[1] == new_point[1]:
            # can't insert a lbrace once a rbrace is inserted
            insert_lbrace_disabled = 1
```

```
            del stack[-1]
        if a[1] < new_point[1]:
            # only insert next block if the previous block is one size smaller
            assert(new_point[1] == a[1] + 1)
            if not insert_lbrace_disabled:
                # print('insert lbrace at {}'.format(new_point[0]))
                result.append(('{', new_point[0]))
            break;
    stack.append(new_point)
    # print(stack)
    # print("\n")

def remove_comments(content):
       def replacer(match):
              s = match.group(0)
              if s.startswith('/'):
                     return " "
              else:
                     return s
       comment_pattern = re.compile(
              r'//.*?$|/\*.*?\*/|\'(?:\\.|[^\\\'])*\'|"(?:\\.|[^\\"])*"',
        re.DOTALL | re.MULTILINE
       )
       return re.sub(comment_pattern, replacer, content)

def preprocess(filename, fileout):
       """
       input: filename to be preprocessor
       output: filename.out file with braces and semicolons inserted at the end of
statement
       """
       f = open(filename,'r')

       content = f.read()

       content = remove_comments(content)
       f.close()
       content = re.sub(r'\n\s*\n', '\n', content)
       content = re.sub(r' *\n', '\n', content)
       ## insert semicolons
       # insert a new line to help with preprocesinng. this is removed before writing
the result
       content = content + "\n"
       p = re.compile("[^\s:;]\n")
       result = []
       for a in p.finditer(content):
```

```
            # print(a.end())
            result.append((';',a.end()-1))
        content = insert_mult_char(content, result)

        # insert braces around conditional statements and functions.
        p = re.compile("\n *")
        stack = [('dummy',0)]
        tab = 4
        result = []
        for a in p.finditer(content):
            assert((len(a.group())-1) % tab == 0)
            new_point = (a.start(), (len(a.group())-1)//tab)
            update_stack(stack, new_point, result)
        f = open(fileout, 'w')
        content = insert_mult_char(content, result)
        content = content[:-1]
        f.write(content)
        f.close()

files = sys.argv[1:]
preprocess(files[0], files[1])
```

queue.c

```
/* Queue functionality for Theatr compiler
Author: Suraj Keshri
 */
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include "queue.h"

head *initialize_queue() {
  int ret;
  head *qhead = (head *)malloc(sizeof(head));

  qhead->queue = NULL;
  qhead->count = 0;

  ret = pthread_mutex_init(&qhead->lock, NULL);
  if (ret)
    perror("Error in initializing the queue lock");
```

```
  ret = pthread_cond_init(&qhead->count_cond, NULL);
  if (ret)
    perror("Error in initializing the count wait variable");
  return qhead;
}

void enqueue(head *qhead, message_t message) {
  queue_t *new_queue = malloc(sizeof(queue_t));
  if (!new_queue) return;

  new_queue->message = message;
  pthread_mutex_lock(&qhead->lock);
  qhead->count++;
  if (qhead->count == 1)
    pthread_cond_signal(&qhead->count_cond);
  new_queue->next = qhead->queue;
  qhead->queue = new_queue;
  pthread_mutex_unlock(&qhead->lock);
}

message_t dequeue(head *qhead) {
  queue_t *current, *prev = NULL;
  message_t retmessage;

  pthread_mutex_lock(&qhead->lock);

  while (qhead->count == 0){
    pthread_cond_wait(&qhead->count_cond, &qhead->lock);
  }

  current = qhead->queue;

  while (current->next != NULL) {
    //    printf("curr->next is null, stepping...\n");
    prev = current;
    current = current->next;
  }

  retmessage = current->message;
  free(current);

  if (prev)
    prev->next = NULL;
  else
    qhead->queue = NULL;
```

```
  qhead->count--;
  pthread_mutex_unlock(&qhead->lock);
  return retmessage;
}
/* we are not deallocating the queue. The programmer is supposed to take care of
that. */
```

- queue.h

```
#ifndef QUEUE_H
#define QUEUE_H
#include <pthread.h>


typedef struct message {
  int val;
  void *argumentStruct;
  void *sender;
} message_t;

typedef struct queue {
  struct queue *next;
  message_t message;
} queue_t;

typedef struct head {
  queue_t *queue;
  int count;
  pthread_cond_t count_cond;/*used to signal count = 1 after count goes to 0*/
  pthread_mutex_t lock;
} head;

head *initialize_queue();

void dealloc_queue(head *);

void enqueue(head *qhead, message_t message);
```

```
message_t dequeue(head *qhead);

/* void print_list(head *qhead); */

#endif
```

filedwld.c

```
/* File downloader library API for Theatr
Author: Suraj Keshri
 */

#include <stdio.h>
#include <string.h>
#include <curl/curl.h>
#include "filedwld.h"

int geturl(char *url,  char *outfname)
{
  CURL *curl;
  CURLcode res;
  FILE *fp;
  // char outfname[] = "www.cs.columbia.edu/~sedwards/classes/2016/4115-
fall/ocaml.pdf";
  // char *outfname = url;
  curl = curl_easy_init();
  if(curl) {
    fp = fopen(outfname, "w");
    curl_easy_setopt(curl, CURLOPT_URL, url);
    /* example.com is redirected, so we tell libcurl to follow redirection */
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp);
    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    /* Check for errors */
    if(res != CURLE_OK)
```

```
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));
    /* always cleanup */
    curl_easy_cleanup(curl);
    fclose(fp);
  }
  return 0;
}
```

filedwld.h

```
int geturl(char *, char *);
```

theatr.ml

```
(* Top-level of the Theatr compiler: scan & parse the input,
   check the resulting AST, generate LLVM IR, and dump the module

Authors:
Betsy Carroll
Suraj Keshri
Mike Lin
Linda Orgeta
*)

type action = Ast | LLVM_IR | Compile

let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [ ("-a", Ast);     (* Print the AST only *)
                              ("-l", LLVM_IR);  (* Generate LLVM, don't check *)
                              ("-c", Compile) ] (* Generate, check LLVM IR *)
  else Compile in
  let lexbuf = Lexing.from_channel stdin in
  let ast = Parser.program Scanner.token lexbuf in
```

```
  Semant.check ast;
  match action with
    Ast -> print_string (Ast.string_of_program ast)
  | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate ast))
  | Compile -> let m = Codegen.translate ast in
    Llvm_analysis.assert_valid_module m;
    print_string (Llvm.string_of_llmodule m)
```

run.sh

```
#!/bin/sh

# Regression testing script for Theatr
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test
#
# Authors:
# Betsy Carroll
# Suraj Keshri
# Mike Lin
# Linda Orgeta



PRE="./preprocessor.py"
THEATR="./theatr.native"
LLI="lli"
LLC="llc"
CC="cc"

Run() {
    eval $* || {
       SignalError "$1 failed on $*"
       return 1
    }
}
```

```
basename=`echo $1 | sed 's/.*\\///
                            s/.th//'`

Run "$PRE" $1 "${basename}.temp" &&
Run "$THEATR" "<" "${basename}.temp" ">" "${basename}.ll" &&
Run "$LLC" "${basename}.ll" ">" "${basename}.s" &&
Run "$CC" "-pthread" "-o" "${basename}.exe" "${basename}.s" "queue.o" "filedwld.o"
"-lcurl" &&
Run "./${basename}.exe"
```

testall.sh

```
#!/bin/sh

# Regression testing script for Theatr
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test
#
#
# Authors:
# Betsy Carroll
# Suraj Keshri
# Mike Lin
# Linda Orgeta

PRE="./preprocessor.py"
THEATR="./theatr.native"
LLI="lli"
LLC="llc"
CC="cc"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0
```

```
keep=0

Usage() {
    echo "Usage: testall.sh [options] [.mc 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=`echo $1 | sed 's/.*\\///
                            s/.th//'`
    reffile=`echo $1 | sed 's/.th$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe
${basename}.out" &&

    Run "$PRE" $1 "${basename}.temp" &&
    Run "$THEATR" "<" "${basename}.temp" ">" "${basename}.ll" &&
    Run "$LLC" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-pthread" "-o" "${basename}.exe" "${basename}.s" "queue.o"
"filedwld.o" "-lcurl" &&
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status an"${basename}.temp"d 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/.th//'`
```

```
    reffile=`echo $1 | sed 's/.th$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""
    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    Run "$PRE" $1 "${basename}.temp" &&
    RunFail "$THEATR" "<" "${basename}.temp" "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`

if [ $# -ge 1 ]
then
    files=$@
else
```

```
    files="tests/test-*.th tests/fail-*.th"
fi

if [ ! -f filedwld.o ]
then
    echo "could not find filedwld.o"
    echo "try \"make filedwld\""
    exit 1
fi

for file in $files
do
    case $file in
       *test-*)
           Check $file 2>> $globallog
           ;;
       *fail-*)
           CheckFail $file 2>> $globallog
           ;;
       *)
           echo "unknown file type $file"
           globalerror=1
           ;;
    esac
done

exit $globalerror
```

git log

```
      commit 7f53c658e082495f0a0751fdada8e21a1ecee5c2
      Author: Beatrix Carroll <beatrixcarroll@gmail.com>
      Date:   Wed May 10 20:08:30 2017 -0400

          added test for recursive function

      plt@ubuntu-plt:~/theatr$ git log
      commit 4a7d3a4bd9358d3868df19d7c7bef0699443d483
      Author: mesamason <mbl2109@columbia.edu>
      Date:   Wed May 10 09:17:14 2017 -0400

          got rid of 2 rules not being reduced

      commit 31c14132729ee0fe4f34fdabed5b88fe6aa256f3
```

```
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 10 07:03:27 2017 -0400

    cleaned as many warnings as I could

commit c71bfe50e5163f7c008b63a7f194a89e3251244b
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 10 06:51:27 2017 -0400

    cleaned warnings

commit c797fcbac0b671d4398289753314ee054977a021
Merge: ba12363 3ea5bb3
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed May 10 06:26:05 2017 -0400

    Merge pull request #28 from mesaMason/warnings

    Warnings

commit 3ea5bb3d2d58a07d654c7d8f43186bbf1574ebdc
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed May 10 06:20:22 2017 -0400

    continued addressing warnings

commit c3082d60c043ba72ae070b234aa17a3769c902b1
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed May 10 05:55:22 2017 -0400

    Started addressing errors

commit 5c5981a9f8731f5b43d55324b77968884ce2730a
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed May 10 05:14:46 2017 -0400

    fixed file structure changes from merge

commit fe958a4ea2aa0eacefd79c9a2e9097a594935833
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed May 10 04:58:28 2017 -0400

    moved files to src/

commit ba1236392b4522e3ab9d952172b116c902a3bbca
Merge: e5d407c 04a21e7
```

```
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Wed May 10 04:27:57 2017 -0400

    Merge pull request #27 from mesaMason/demo-and-if-blocks

    Demo and if blocks

commit 04a21e71f09ac0210aa0564d94f1b71b765db445
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 10 04:16:22 2017 -0400

    switchboard demo done

commit 89c44b2f6f7ac46f4ddd132f9c1cccb256d092ac
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 10 02:40:05 2017 -0400

    demo started and working on fixing if statements in actors

commit e5d407ce63473ffeaf586dd279f7f0e838ce908d
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 03:50:41 2017 -0400

    adds a run script. modifies prints in single actor download code

commit 32cf20919be6aa6416e7d787aaa52dfb39c0391e
Merge: 36e0bb4 861f013
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 02:10:08 2017 -0400

    Merge branch 'master' of github.com:mesaMason/theatr

commit 36e0bb41a0321cef5f59dd43ebcb684437687fa3
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 02:07:40 2017 -0400

    fixes concurrent download actor. adds a single actor download

commit 861f0135559dbaa25ee198f2229dd3cd01db98a5
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 02:07:40 2017 -0400

    Fixes issue in test-actor-download.th

commit 760886eae780f0065952d4db869ea0ef4487cd3b
Merge: a0c1511 e3cf741
```

```
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 00:05:57 2017 -0400

    Merge branch 'master' of github.com:mesaMason/theatr

commit a0c1511b9f7c8122729108e970476d6288a892f4
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed May 10 00:05:54 2017 -0400

    adds a download actor demo in tests directory

commit e3cf7415a05d3246742bfbf6d2a9188eaf1aed33
Merge: 6329257 c0841a8
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed May 10 00:03:04 2017 -0400

    Merge pull request #26 from mesaMason/strongly-typed-send

    Strongly typed send

commit c0841a8d896287103ec241b1b47a8cfd750d6791
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Tue May 9 23:58:54 2017 -0400

    changed actor send syntax to strongly typed

commit e33e17a34a4d7dea73ae03ea0502908069da0154
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 23:26:33 2017 -0400

    strongly typed send working

commit 63292572f3381cda87c84656a62988912018dc65
Merge: 8f7c167 364ded7
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue May 9 22:14:47 2017 -0400

    Merge branch 'master' of github.com:mesaMason/theatr

commit 8f7c167a0ff059d3c414c019744c9c2b4f6bc564
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue May 9 22:14:44 2017 -0400

    Adds geturl function, removes appending newline to all the stringlits

commit 364ded7f26d87976cf6a5d9e6f18f5aa1d700d99
```

```
Merge: 22faa8f c435ff7
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Tue May 9 20:17:08 2017 -0400

    Merge pull request #25 from mesaMason/BC_fixing_strings_bugs

    Bc fixing strings bugs

commit c435ff7350e1e0b5089144493c99c30eed5108dc
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue May 9 19:19:32 2017 -0400

    fixed scanner

commit 0f0ce8d7ec3bee71d7170cb938748baa581b8753
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue May 9 19:14:58 2017 -0400

    fixed tests, threw out garbage test files in master

commit 22faa8f762f2e0975bfeb3a6fe51f8d1fb4d1683
Merge: 4f04c27 e6e7d33
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Tue May 9 17:07:40 2017 -0400

    Merge pull request #24 from mesaMason/msg-queue-combined

    Msg queue combined

commit e6e7d33455741829999098cf6e60effefda0e5cd
Merge: 699a0e9 4f04c27
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Tue May 9 16:24:49 2017 -0400

    Merge branch 'master' into msg-queue-combined

commit 4f04c27fb4d55763dd5d2fe6da515ac6c5cfdc30
Merge: e49cba5 f43a33a
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Tue May 9 16:00:47 2017 -0400

    Merge pull request #23 from mesaMason/BC_adding_string_variables

    string variables are working, but I am not sure if escaping has desir…

commit 699a0e9a4c87d1537ffa58574eb41cfb3d836d66
```

```
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 15:59:48 2017 -0400

    fixed tests so actors get sent a die() message, so won't loop infinitely

commit 889e9174ca89adfbc8b9d44eaeb21df9e898b10d
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 15:25:50 2017 -0400

    WORKING MULTIPLE MESSAGES PASSINGgit statusgit status still have lots of
print statements to get rid of

commit bf7728c38bb8874a2b9655b4b8a1f2ae8dfcfe8b
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 14:52:23 2017 -0400

    dequeue still not being passed the same head* as enqueue and initialize
queue

commit fbec86d524ea4e460d6a98e821369b639885e30e
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 11:05:06 2017 -0400

    got enqueueing working but need to fix the return type/value of dequeue

commit 2f8569ce2e9d5ac80e96f63364bd11765a31201d
Merge: 9b1e998 bf00b4f
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 03:49:26 2017 -0400

    merge

commit 9b1e99865edfaee1232b7f1be7f04fcb7a400131
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 03:46:26 2017 -0400

    no errors but enqueuing not happening

commit bf00b4f7ebfddac4eb05805e34ff19e04611dbbc
Merge: 09f635f 4dc3419
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Tue May 9 02:37:08 2017 -0400

    Resolved merge conflicts

commit 09f635fb0317020d030f34ccf96bba2a9d566499
```

```
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Tue May 9 02:23:50 2017 -0400

    Commented msg testing hack causing most of seg faults. Msg while loop
needs to be tested. Sending throws seg fault

commit c5327c4520707ba8c267e26fca36cc3534fa3fbc
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 01:59:19 2017 -0400

    linking working but lots of segfaults

commit f00ed39438424d4966db4573819ea6280f14f428
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 01:28:21 2017 -0400

    got receive functions pulling arguments with no errors, need to actually
test it

commit f2ec86fc1aeb452f35e4e709728f65d6a1192e78
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 23:47:32 2017 -0400

    Working on passing actuals struct to msg

commit 202b85da6b592806e62869663b60a34df967c3b2
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon May 8 22:02:18 2017 -0400

    connected send to receiver type to turn a receive function name to a case
number

commit a91fe42425dda6d8d4a5a60118149a66d5f5f4c4
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon May 8 19:04:26 2017 -0400

    fixed instruction is not an instruction error, but linker still not
working properly

commit 80ccc3b000ead969fe0accedcd95fcd0cf20a5db
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon May 8 17:54:09 2017 -0400

    added linker

commit 7ea8ffce4f432e231c7b97dd23c1fa11fce6eeaf
```

```
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 16:11:11 2017 -0400

    fixed parsing error for one of tests

commit 286190b05ae318f3a3694067dcaa289a0ef288c2
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 15:59:32 2017 -0400

    msgs now have access to their msg_struct. Added msg formal args tests.

commit f22d2d780d4d64e71686531b4a997b1b5b84b22b
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 12:57:13 2017 -0400

    Actors with formal args msgs compile

commit d3bc9f6bdc2ae3e6eed84f1cdadbcf05874ae49e
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:08:51 2017 -0400

    resolved merge conflict

commit ca6785df7ccedfab0c8d295be7157b0823bc6e3a
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:06:30 2017 -0400

    Resolved merge conflict local vars

commit 4d9d4db0201544d1970806a21f1b37152cbe9122
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Sun May 7 16:54:41 2017 -0400

    Added tests for testing msg's ability to change actor's formal and local
variables. Passes in codegen msg test mode. See build_body_block comments.

commit 07322c79ca5a03651ea52f0b4bee6f99a5487ce0
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:05:12 2017 -0400

    Resolved merge conflict

commit 6f41f5d848090705451e0976baa2a18ce51fc233
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:02:20 2017 -0400
```

```
    resolved merge conflict

commit bc41f72b180ce3863f8b8583d0821c4d7858583f
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Mon May 8 15:53:10 2017 -0400

    Adds llvm code for queue. Compiles but run time error.

commit b036d85d1975af4a5c4cc65fa800f5e3884246d2
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sun May 7 13:43:41 2017 -0400

    Adds llvm code to initialize various global functions and variable for
calling queue related functions. Allocates the queue on the heap

commit 4dc341948fe5f10509f0373885b4f3563216b049
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 01:59:19 2017 -0400

    linking working but lots of segfaults

commit bab43e5da914f26ba0da3c5880634f1351de42cc
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 9 01:28:21 2017 -0400

    got receive functions pulling arguments with no errors, need to actually
test it

commit 6a1259f0d5fccbe2e261cde136ab1ed6656f3654
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 23:47:32 2017 -0400

    Working on passing actuals struct to msg

commit 33c1c7337e2524c805201bf301bdd68f930fcfe5
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon May 8 22:02:18 2017 -0400

    connected send to receiver type to turn a receive function name to a case
number

commit f9259bbf5d50b79c85cd7e9c6ff5128284d073b9
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon May 8 19:04:26 2017 -0400

    fixed instruction is not an instruction error, but linker still not
```

```
    working properly

    commit ecc496ac733af52a7a833dae012230fa59573452
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Mon May 8 17:54:09 2017 -0400

        added linker

    commit 1e59e6442e72176823e95f864313459b39e6c439
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Mon May 8 16:11:11 2017 -0400

        fixed parsing error for one of tests

    commit ad3c7e0140a0327ee954374eb90769422fedc88d
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Mon May 8 15:59:32 2017 -0400

        msgs now have access to their msg_struct. Added msg formal args tests.

    commit 0aae95185a0beaa0d166aaa318a550ac922debad
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Mon May 8 12:57:13 2017 -0400

        Actors with formal args msgs compile

    commit bc0795209439de6ae762bc6a27bf37db0a996ea0
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Mon May 8 17:08:51 2017 -0400

        resolved merge conflict

    commit 2bbe66fae2db7b4719cf6be9e91551f243c21ee3
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Mon May 8 17:06:30 2017 -0400

        Resolved merge conflict local vars

    commit 563a73b21fe48156f6f58300e574c7885b0d0b91
    Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
    Date:   Sun May 7 16:54:41 2017 -0400

        Added tests for testing msg's ability to change actor's formal and local
    variables. Passes in codegen msg test mode. See build_body_block comments.

    commit 995c0ba4af67f1676b1ef059aa0674927e43ecdc
```

```
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:05:12 2017 -0400

    Resolved merge conflict

commit db362e0ff3db820188230864140f791625f134ad
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Mon May 8 17:02:20 2017 -0400

    resolved merge conflict

commit 332fb8b0dedfe646314a895d687ac3620da201d6
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Mon May 8 15:53:10 2017 -0400

    Adds llvm code for queue. Compiles but run time error.

commit f43a33ab3918e9f3a6febcdce3e694f30489233e
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Mon May 8 12:20:58 2017 -0400

    Still dont have escape chars working in strings

commit ba50646ad6eb446b3ddeeb8b9d5823c713cf6aa9
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Mon May 8 01:40:54 2017 -0400

    string variables are working, but I am not sure if escaping has desired
behavior

commit e3aa5573a7f2a5717dba441158efb7898814390a
Merge: e49cba5 5e0a3fb
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sun May 7 13:44:41 2017 -0400

    Merge branch 'SK_message_llvm'

commit 5e0a3fb3f9a4c0d1aeb90b02e1e3f2a3e01c0fa5
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sun May 7 13:43:41 2017 -0400

    Adds llvm code to initialize various global functions and variable for
calling queue related functions. Allocates the queue on the heap

commit e49cba5c18a29c3e7d80e18bacdef783f53c3335
Merge: 255d6f7 e61592b
```

```
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Sun May 7 00:14:43 2017 -0400

    Merge pull request #22 from mesaMason/ML_message_passing_parsing

    Ml message passing parsing

commit e61592b9103e016d9bd7c8b75e5a16745afadae0
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 23:49:45 2017 -0400

    added test to create actor in actor

commit b49cf74436609c373209a3426f588f6717c1038e
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 23:47:09 2017 -0400

    global array of actor_address_struct working, main() function calls join
in a for loop over all tids. So now we can create actors inside of actors

commit d50d318334e9fee53072cd98cbc057e78c3e839c
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 23:07:06 2017 -0400

    global array as pointers now but having trouble dereferencing when
pthread_join is called. going to try to change global array to hold the actual
address structs instead of pointers

commit a2f46cf5cfd8f315d7578eae57bced832b8c60ab
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 19:24:56 2017 -0400

    added c example

commit 0b893f6040bd7dcfbe44ca7d3e645eecd43cfbec
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 16:28:29 2017 -0400

    created global array to keep track of created actors

commit d29e227de72dbee1d393f1f2fc82581604a7466d
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 13:44:53 2017 -0400

    sending parsing and scanning working
```

```
commit 255d6f7f5f1acea71d980556af94f59381afc867
Merge: ee7a4a1 4f8beea
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Sat May 6 23:48:24 2017 -0400

    Merge pull request #21 from mesaMason/BC_adding_tests_all_around

    Bc adding tests all around

commit 4f8beea0921b179acbc27e59f579458513c512e2
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 22:50:54 2017 -0400

    added some tiny fixes

commit fff17c8116aa9bbc180b75354c5b7aafbc6426e0
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 22:48:13 2017 -0400

    adding more tests for existing stuff

commit ee7a4a13398547b34e6b97c915597f08b0604f1d
Merge: 2020943 86a7db7
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Sat May 6 20:26:40 2017 -0400

    Merge pull request #19 from mesaMason/ML_switch_pull_msg_info

    added codegen to access the pointer to the msg_struct before the swit…

commit 202094346b815412d4bc6ac0552c10911f591ed7
Merge: 669f5e7 9d4651d
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Sat May 6 20:26:24 2017 -0400

    Merge pull request #20 from mesaMason/BC_adding_comments

    Bc adding comments

commit 9d4651dfe6e6f1a6dc70945b559988d46d587f07
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 20:00:55 2017 -0400

    forgot to take out garbage comment done now

commit 6a172d31b0037a505b5f8d0f55fc35692f1bf680
```

```
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 19:59:02 2017 -0400

    added comments ability to preprocessor

commit 643061f9accd668870a1f76ab591f6a4d244ce5e
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 19:58:40 2017 -0400

    added single line comments to scanner

commit 52df146dab75af1a8c423c4e00329972328acbf7
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Sat May 6 19:58:21 2017 -0400

    added tests for single and multi line comments

commit 86a7db718b7fc9125843472bc54383fb5f9b250d
Author: mesamason <mbl2109@columbia.edu>
Date:   Sat May 6 12:59:39 2017 -0400

    added codegen to access the pointer to the msg_struct before the switch
statements

commit 669f5e7c0cf2260a209c9ba59ec00feaf2d355fa
Merge: 6ca99e6 3ce8905
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat May 6 14:06:44 2017 -0400

    Merges struct declaration and message queue c code

commit 3ce89050dbee05682bf4c7d9193b9dd0de3b243c
Merge: c130ff4 41f3a8e
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat May 6 01:14:25 2017 -0400

     fixes merging errors

commit c130ff485df73f5aef92c4430a4eb9b8fb1e3576
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat May 6 01:11:51 2017 -0400

    Removed few comments

commit ad1ab61446619ef9acbb0f1ac106ee007fd6b7f8
Author: Suraj Keshri <skk2142@columbia.edu>
```

```
Date:   Sat May 6 01:04:26 2017 -0400

    Message passing works with multiple threads sending message and one thread
listening

commit 41f3a8ee4d543890708ad827f0054c96cc30625b
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat May 6 01:04:26 2017 -0400

    Message passing works with multiple threads sending message and one thread
listening

commit 6ca99e6f499e8277712c040cff125b82347a02d0
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri May 5 16:37:41 2017 -0400

    modified add_formals function to be generic

commit c4cd96c8e5dcb0442a3f38c0228f813d5980bb1b
Merge: a9bdac0 e5cfdea
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Fri May 5 15:29:58 2017 -0400

    Merge pull request #18 from mesaMason/actor_codegen

    Actor codegen

commit e5cfdeaac5f5e5601378fe5d339ba4ae52d1e09d
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri May 5 15:28:03 2017 -0400

    added switch c example

commit b2cb364292288c185f45586a14a8c77396f9f5ca
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri May 5 14:56:46 2017 -0400

    added while loop for messages

commit 19509c62ecc5fb8d69f78496cade28dff42ac7ec
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri May 5 13:39:27 2017 -0400

    added threads with global array example

commit a9bdac025b1e214e352a3243c36bb071ee74eef7
```

```
Merge: 5fce48d b396ca7
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Fri May 5 12:24:43 2017 -0400

    Merge pull request #16 from mesaMason/actor_codegen

    Actor codegen

commit 5fce48d7785dd14036341b4e93035e563f1c7db1
Merge: 9c3c282 2b66706
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Fri May 5 12:17:58 2017 -0400

    Merge pull request #17 from mesaMason/BC_logical_ops

    added tests for logical operators

commit 2b6670646ab456b9705c74d3a75f192876a57fc2
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Fri May 5 12:11:12 2017 -0400

    added tests for logical operators

commit 0a99a0f2b0f3ee7d27fc1df4eba8100761e7baa1
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Fri May 5 01:23:34 2017 -0400

    Adds conditional wait variable for enqueue/dequeue. This ensures that the
actor would wait on the queue to be nonempty before it can dequeue a message
off it

commit b396ca787e7b7f2f1b42522b70a4bc25a894172e
Author: mesamason <mbl2109@columbia.edu>
Date:   Thu May 4 23:15:04 2017 -0400

    refactored code - brought expr builder OUTSIDE of both build_function_body
and build_actor_function_body functions, and replaced local_vars StringMap
with a ref StringMap

commit 445b745cc4b8bb82d647296e5c1ab72728c76fd7
Author: mesamason <mbl2109@columbia.edu>
Date:   Thu May 4 19:25:44 2017 -0400

    actors pthreads join if created in the main() function, but nowhere else

commit a8236084b071032f2e8debadc376ca2fbc46e68b
```

```
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 3 22:29:15 2017 -0400

    experimenting with adding pthread_join to end of main by keeping a global
list of active_tids

commit b340bbaa8e79966c61941569831a5235820d6498
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 3 17:08:53 2017 -0400

    actor instantiation working, using malloc

commit 9c3c282ab1262a767e79355f70a9c7dadee8edd3
Merge: f0e5c09 2149d9d
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Thu May 4 22:59:39 2017 -0400

    Merge pull request #15 from mesaMason/BC_trying_to_fix_boolean_bug

    Bc trying to fix boolean bug
    Sorry just saw this now, looks good!

commit 046de439ed19211023993c165ffdc8e9df926136
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu May 4 19:39:32 2017 -0400

    C prototype for actor calling function dynamically

commit 2149d9d02efeb8d313fa9e177047e7a775141877
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu May 4 15:40:09 2017 -0400

    changed print funcion to print 0 and 1s for booleans instead of true or
false

commit c7eb875ff73c5b1b03a3d08f8fcbf7e441a6567b
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu May 4 15:37:18 2017 -0400

    fixed test-print now that we are using 0 and 1 for boolean

commit 17530612f785bbb16dd0cd2dc4976bc57a92bc8d
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu May 4 15:36:43 2017 -0400

    adding tests for comp that adhere to 0 and 1 now
```

```
commit 319223f047fc213a922c2814be8e3395382eefec
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu May 4 15:31:47 2017 -0400

    added to clean in make

commit 45a6c1b604ceade3df4cdb6cd0c42848c6c00f78
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu May 4 15:31:16 2017 -0400

    fixed tests for comp float for new boolean

commit 854ee7f04fc83c42da5be4e9a5de6aca927343bb
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu May 4 14:58:13 2017 -0400

    Adds locking to queue

commit ec53cfaa3263e44c98c354f338e6d1b96dd65954
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu May 4 14:31:09 2017 -0400

    Fixes seg fault. Adds a list declaration test

commit e99d750609add3e1f4ad1ddfbf95b37e4e29fcca
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu May 4 02:58:05 2017 -0400

    Parses struct. Adds queue code in queue folder, seg fault while dequeing

commit f0e5c09639c0994ffb105226b5c77c65859f91ae
Merge: 356787f 02e2320
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Wed May 3 13:35:31 2017 -0400

    Merge pull request #14 from mesaMason/actor_codegen

    Actor codegen

commit 02e232051ce8d9c2d48b39c7ed4be3a6a36d870a
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 3 13:09:02 2017 -0400

    before rebase
```

```
commit a59dee9511d7fb7b0a01a77b5f2b1a87b9c450d6
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed May 3 13:03:57 2017 -0400

    codegen done for new actor, but need to move state struct from stack to
heap

commit d903c8fca3a7ecd01dd1589978c91e0071753480
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 2 22:03:27 2017 -0400

    got struct and local var generation working? need to add codegen for new
actor instantiation to test it!

commit 9628203d868faa4683e3fb28928b7e57f4fee2e8
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 2 21:20:13 2017 -0400

    drafted codegen for creating the struct and local variables, not working

commit 6855a5fe681cc46de5ad8b3785c2c47097a2a4ff
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 2 17:31:45 2017 -0400

    changed actor decl syntax to allow statements before receive instead of
just variable declarations - makes declaring local state variables possible

commit 6a9171ae92d1c5bb273cf91736c81fbb4c277967
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 2 15:06:10 2017 -0400

    started codegen generating local variables for actor's pthread function

commit bcb41df8c70d2d9922fe90ba7284b189051c9f5f
Author: mesamason <mbl2109@columbia.edu>
Date:   Tue May 2 14:09:41 2017 -0400

    added c examples to understand llvm

commit b7d559c49d0eba5e9999b337859418c7db286a0a
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri Apr 28 17:13:07 2017 -0400

    got passing struct to pthread working

commit 06287b1b2a0fa6455a7762d9c830a9ebfaf96e35
```

```
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Thu Apr 27 23:07:45 2017 -0400

    LO: struct compiles, but prints pointers of data passed in

commit d9e7c85d6e7ba6fae93f709b25bcf1bdaa1632b4
Author: mesamason <mbl2109@columbia.edu>
Date:   Thu Apr 27 20:53:45 2017 -0400

    bitcasting not working on the struct

commit 96dfc9e1ac98b836361ed8929299d4dc78bfd391
Author: mesamason <mbl2109@columbia.edu>
Date:   Thu Apr 27 19:10:24 2017 -0400

    undeclared variable calc error

commit 1cead646a0424339c1458ab443552ec23a1bfe04
Author: mesamason <mbl2109@columbia.edu>
Date:   Thu Apr 27 17:12:41 2017 -0400

    structs not yet working but struct type created

commit 356787fb97dd5a746aef102632735d9aabb9b666
Merge: df0b716 423994c
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Sat Apr 29 13:12:35 2017 -0400

    Merge pull request #13 from mesaMason/BC_adding_float_REBASED_but_has_bugs

    Bc adding float is now done

commit 423994cdb797ba3315b3b496b36996d59a1830cb
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat Apr 29 13:06:44 2017 -0400

    fixes ast.ml bugs

commit 9f12f60c1339bcceb15392b2e40349b377a6dc33
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu Apr 27 17:10:02 2017 -0400

    fixed comments in cg

commit 8110a7d919d2816b1fe760b15f8b38bc3bf5d995
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
```

```
Date:   Wed Apr 26 19:46:31 2017 -0400

    fixed ast to have actor for string typ

commit bfd7470fc459dec083bddc637b4c31e5d184a308
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Wed Apr 26 18:54:24 2017 -0400

    float arith and comps are all working, but float comps seem to only work
on float literals, not on variables that are floats. Going to rebase off
master before proceeding

commit a8b432d41f996cdd3146e6161c4ff4ed8c9454a8
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Wed Apr 26 13:22:49 2017 -0400

    started trying to get comparison to work for floats

commit 65be19f6fd56b290a6151d9c5dcd9639951cecf2
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Wed Apr 26 11:37:11 2017 -0400

    arith ops are all working for into and float. Now going to make comparison
ops work for float as well

commit 12553e238c67a27cb912809b914bad894330f450
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Wed Apr 26 11:33:47 2017 -0400

    changed my change in makefile to match the format of lindas to avoid any
conflict

commit b750191577c7a158a870c3e0e046955fb7841742
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 25 13:05:29 2017 -0400

    started refactoring codegen to pull out decisioning for binops for various
types into a separate handler method

commit 18b04be16da3167adafcbaf109a98f5a94d30861
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 25 12:53:08 2017 -0400

    fixed the test-arith-binops file, even tho its redundant now i guess

commit 145536a86383cb7a5f2fb2d05454784996501818
```

```
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 25 12:50:55 2017 -0400

    added diff test files for int and float binops

commit eb431ba78c000ea7fcd83aa86ba1be91ae8e9399
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 25 11:41:03 2017 -0400

    float is working for adding. Need to refactor code gen to make float work
for the other ops now

commit bcaae8324b5a53b06c41d72ce64ffcb2063b3b9b
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Fri Apr 21 16:59:35 2017 -0400

    added test for arith_binops, float stuff not passing yet

commit b9c034c9efeec69dfd47abfe60a2a0a19a2f0b4f
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Fri Apr 21 16:58:19 2017 -0400

    added Double to print_typs

commit 252805693e18971b9d46696c3808591de60b068b
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Fri Apr 21 16:49:01 2017 -0400

    changed the naming of float stuff in codegen in the hopes of fixing it

commit 1d66b6a94fb1fe2ace4a1b1c7ab0283bd0c7de15
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Fri Apr 21 16:44:35 2017 -0400

    fixed Make clean

commit d57a2f96b9202531bf800e2edbf476ecc3b414d8
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 11 17:13:28 2017 -0400

    changed semant.ml for double, but not sure if the syntax is right for my
ocaml

commit 72ab3e97f566c40421f102804e49ffd63b69319b
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 11 14:07:36 2017 -0400
```

```
    added changes for double implementation

commit df0b716a188e1a44ab276abbb43b60e66ba4344d
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Thu Apr 27 18:01:08 2017 -0400

    BC: added test cases for comparison that now fail. == etc has not worked
with any declared variables. only with literals. was not discovered til today.
floats was not the issue

commit db4c7f2e3abe56c94625debfdbff5c6dfd36b9b8
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu Apr 27 01:00:22 2017 -0400

    Parses collection type. One test added. Code generation is not implemented
yet.

commit 54ee37186984df14dd36ecc7cfcbce74ba3c958f
Merge: 338e725 14869cd
Author: mesaMason <mesaMason@users.noreply.github.com>
Date:   Wed Apr 26 11:41:13 2017 -0400

    Merge pull request #12 from mesaMason/actors-llvm

    Added pthread function for funcs with 0 args or 1 int arg

commit 14869cd23c60d79e937cd07c48b4ba8a0fa368fe
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Apr 26 02:38:53 2017 -0400

    Resolved merge conflict

commit 38d0d55e6a7928a80fd547143806de8158b89fb7
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Apr 26 02:04:22 2017 -0400

    Added pthread function for funcs with 0 args or 1 int arg

commit 338e7256eb70a27a39effd74c422ffadbda28be0
Merge: d72c497 2e0dc48
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 25 17:49:03 2017 -0400

    Merge pull request #11 from mesaMason/ML_actor_instantiation
```

```
        working syntax and semantic checking for new actor instantiation

    commit 2e0dc489aa2ee571ec5e99fa4722994609d26d28
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Tue Apr 25 14:58:02 2017 -0400

        working syntax and semantic checking for new actor instantiation

    commit d72c4978291ac5ad3835536cc15c3e6db387b1be
    Merge: 220f50b 1ad3317
    Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
    Date:   Tue Apr 25 17:32:16 2017 -0400

        Merge pull request #10 from mesaMason/SK_ML_interleaving_global

        working variable declaration and definition

    commit 1ad3317d86cc00382221c0088ff42224a0d25a1e
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Tue Apr 25 17:22:18 2017 -0400

        working variable declaration and definition

    commit 220f50b58dafdc87fa43652820578f9023148ddb
    Merge: 18ef4f4 8ae18c6
    Author: mesaMason <mesaMason@users.noreply.github.com>
    Date:   Mon Apr 24 19:36:29 2017 -0400

        Merge pull request #9 from mesaMason/ML_start_actors

        Ml start actors - merging ahead of Suraj and me working on adding
    functionality to declare and initialize variables

    commit 8ae18c6b2cabcd17c81ad1193d75dd2480040c38
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Sun Apr 23 21:29:11 2017 -0400

        strengthened interleaving to check for void variables and redeclaration of
    the same variable

    commit 4539cdaaf9253aa7f858a017ed5c632209270c5c
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Sun Apr 23 19:24:35 2017 -0400

        got interleaved statements and variable declarations working within
    functions
```

```
commit 18ef4f41451ae17c573834aa06b3b4e5669908c5
Merge: 9bff954 9f82622
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Fri Apr 21 16:52:59 2017 -0400

    Merge pull request #8 from mesaMason/ML_start_actors

    Ml start actors

commit 9f82622be9dd76c625c25cb27535d4c9c8940c9c
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri Apr 21 16:46:02 2017 -0400

    added c-examples

commit 7fc2c178226281f705e8537a96a61d13ac51952d
Merge: 0e4b592 c789c37
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri Apr 21 12:26:23 2017 -0400

    Merge branch 'ML_start_actors' of https://github.com/mesaMason/theatr into
ML_start_actors
    weird error?

commit 0e4b592a9810a8bddab3b36a2189c8c37349e61f
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri Apr 21 12:03:11 2017 -0400

    got test-actor-syntax working

commit 10433d9f2d074f5795777700b115e04c5615f706
Author: mesamason <mbl2109@columbia.edu>
Date:   Sun Apr 9 19:16:34 2017 -0400

    finished syntax and parsing for actor declarations

commit 9078c0ff603749bbf97d694d96f2c80f39a1df76
Author: mesamason <mbl2109@columbia.edu>
Date:   Sun Apr 9 17:36:31 2017 -0400

    added actors to ast, parser, scanner

commit c789c37ed06e2e373420664f9802eb3da614290e
Author: mesamason <mbl2109@columbia.edu>
Date:   Fri Apr 21 12:03:11 2017 -0400
```

```
    got test-actor-syntax working

commit 9bff954b6c8a6be9729d00e83072e3f6765827dc
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 11 15:36:53 2017 -0400

    fixed comp test that had wrong syntax

commit 732cb9d5d4d390b2a60d4ac1f961a1423dcaae38
Author: mesamason <mbl2109@columbia.edu>
Date:   Sun Apr 9 19:16:34 2017 -0400

    finished syntax and parsing for actor declarations

commit 232d7a301e3bb0e52ac042d655f6354f13116026
Author: mesamason <mbl2109@columbia.edu>
Date:   Sun Apr 9 17:36:31 2017 -0400

    added actors to ast, parser, scanner

commit 1d47e0bf55adbcc72674401db5d14a91f98e8588
Merge: 2328752 de2118f
Author: Beatrix "Betsy" Carroll <beatrixcarroll@gmail.com>
Date:   Thu Apr 6 15:46:33 2017 -0400

    Merge pull request #5 from mesaMason/BC_adding_comparison

    Bc adding comparison

commit 23287526925146338d53933c7573187993d73440
Merge: e0d5f66 47c1daa
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed Apr 5 15:40:20 2017 -0400

    Merge pull request #6 from mesaMason/small-changes

    Changed Literal to IntLit and created util/

commit 47c1daaf07bd3b73bed4e81442f0e1b0118b1204
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Apr 5 15:38:11 2017 -0400

    Changed Literal to IntLit and created util/

commit e0d5f66f49fd831eab81c7a969de86374307b5a7
```

```
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Apr 4 19:17:25 2017 -0400

    Preprocessor working version. Preprocessor doesn't take care of comments

commit e942a2648d2ebd1f1ba7c01aa1ac8917332fafe7
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Apr 4 18:13:14 2017 -0400

    Modifies print test so that it fails. Updates preprocessor.py

commit de2118f418b9c444483cc39f073c7c6e0de40413
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 4 15:48:35 2017 -0400

    added test for comparison operators

commit d0e083ed5f4037861b707176be762b09d0dc4e07
Author: Beatrix Carroll <beatrixcarroll@gmail.com>
Date:   Tue Apr 4 15:47:46 2017 -0400

    added rm -rf *.ll to make clean, to get rid of the files that failed tests
create

commit bfc31a5af18db1782d6d767ac2fb70a1de9500fd
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Mon Apr 3 18:50:57 2017 -0400

    Adds .out file for test-while2.th

commit 8d3d5596c5e1c0a82985686e254699459146d7f2
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Mon Apr 3 13:57:59 2017 -0400

    Adds another while test that fails.

commit be9d00ea634bfde9de12e7c36aab2d7338b1ae48
Author: skk2142 <skk2142@columbia.edu>
Date:   Sun Apr 2 15:35:23 2017 -0400

    Preprocessor: Adds semicolon at the end of statements. Fixes output file
issue

commit f4ef6c424240985c122eb0c0c9f0af101f30533d
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sun Apr 2 14:09:34 2017 -0400
```

```
    Fixes filename reading issue in  preprocessor.py

commit 470ce1b8b01e70c32278d9fcba69134260956ffa
Merge: a6ba3c2 96c715c
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Sat Apr 1 21:13:22 2017 -0400

    Merge pull request #4 from mesaMason/print-func

    Overloaded print func

commit 96c715c832d213ce21c3077fd5320fbbd1983f53
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Sat Apr 1 21:09:13 2017 -0400

    Overloaded print func

commit a6ba3c2f94a3b6f7277306ea8cc8ff01e07258a5
Author: skk2142 <skk2142@columbia.edu>
Date:   Sat Apr 1 19:42:20 2017 -0400

    adds preprocessor.py

commit 03e8d18ad555c946d093ba451571524da12e262c
Author: mesamason <mbl2109@columbia.edu>
Date:   Sun Mar 26 18:35:34 2017 -0400

    added if, else, for, while in Theatr syntax, with tests

commit 940bb103d9760b66a2677156df445c8ee92f31e7
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Sat Mar 25 18:22:24 2017 -0400

    Working hello world. Edited microc

commit da8cd02e854f3856847a7209083d8213f1d10a2e
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Sat Mar 25 17:17:55 2017 -0400

    Resolved compilation issues

commit ff309beb2bfce6aeba1148fdce786c86f9936824
Merge: 1e623af 4ed7bcc
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Fri Mar 24 18:50:38 2017 -0400
```

```
    Merge branch 'master' of https://github.com/mesaMason/theatr

    Updating master

commit 4ed7bcca37a4e32c3d4f360112209ec781212389
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Sat Mar 25 13:00:13 2017 -0400

    Modifies bind type in Ast.ml. Modifies formal_list in parser.mly

commit 1e623af83cc561e01722ba9c1e06e8f4aafc423e
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Fri Mar 24 18:49:16 2017 -0400

    Moved semand codegen to top-level directory

commit 20ea539feca583f54f1817f2607d8cf2b233f020
Merge: 75ca3f2 9966ca2
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Fri Mar 24 09:54:48 2017 -0400

    Merge pull request #3 from mesaMason/string-lit-microc

    Added printing string literals to MicroC code

commit 9966ca2c6d316b36792eb6a3b2f98112d35029a8
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Fri Mar 24 02:22:58 2017 -0400

    Prints only strings with 'print' function

commit 9c27c7191ac0252ca1e825215c111a5150f16144
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Fri Mar 24 00:00:34 2017 -0400

    Prints string literals without \" and ignores whitespaces following string

commit 91fd2ded647ad7153d1d9845d641d03f0088747d
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Thu Mar 23 23:37:12 2017 -0400

    Prints multiple space-separated string literals and punctions

commit 8cc600bbe0c8ef80c0cb9783399aec06b77ab471
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
```

```
Date:   Thu Mar 23 23:06:59 2017 -0400

    Prints single string literal with newline

commit 75ca3f2ee66addfe1cea21f4dfa6adcc42a1414e
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu Mar 23 22:48:50 2017 -0400

    Fixes some code duplications

commit 753923d91bdbeb78d1e74cd3d7baf4c86e88bb2d
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Thu Mar 23 21:49:05 2017 -0400

     Modifies scanner.mll to make token names match the token names in
parser.mly. Adds grammer for typ in parser.mly

commit ddb47c81e48baf73142e014f0a7a1c7c5210ab04
Author: mesamason <mbl2109@columbia.edu>
Date:   Wed Mar 22 22:15:39 2017 -0400

    parser and ast support MicroC functionality and also string literals,
using Theatr syntax. Tested with ocamlyacc and no shift/reduce conflicts

commit b12d942e37837ce8e585af44bef437d82cef55a1
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Wed Mar 22 14:34:35 2017 -0400

    Modifies fdecl in parser.mly

commit b7057b7f2eed4ac2815bf376d0506408c6760fe3
Merge: 762352a 2c8ec62
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed Mar 22 13:41:47 2017 -0400

    Merge pull request #2 from mesaMason/testing-file-system

    Deleted uncessary files and edited theatr test cases

commit 2c8ec626c4e290ee477e041e37fa2e1f63c30271
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 13:39:02 2017 -0400

    Deleted uncessary files and edited theatr test cases

commit 762352af501fa9e0caa0c1ba315bac5aace26dca
```

```
Merge: 45c27e9 c1ea0c6
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed Mar 22 13:30:41 2017 -0400

    Merge pull request #1 from mesaMason/testing-file-system

    Testing file system

commit c1ea0c6aa0d179d2eb2e99ef5d6f29a9516a6a9c
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 12:53:12 2017 -0400

    Added folder for testing small changes to microC

commit 2efdb2d231fb4770d1cc7d0d5406219bd01852f9
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 00:32:33 2017 -0400

    Reset scanner to Theatr scanner

commit 80d91058a570a341ee55e2909fc77ffa9db841ba
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 00:25:48 2017 -0400

    Added compiled scannerprint to .gitignore

commit 8ceb7f2ac26b96e074786313ecd7ee16a07758c4
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 00:23:38 2017 -0400

    Updated testall.sh file for Theatr

commit dc9f7fe1fe0c056f7b7db87a2fdc2e88c33e35dd
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Tue Mar 21 22:17:25 2017 -0400

    Changed references to microC with Theatr

commit 92c6377e9a80ec830e8ad1ce7e631693e36313ea
Author: Linda Ortega Cordoves <linda.ortega2013@gmail.com>
Date:   Wed Mar 22 13:20:00 2017 -0400

    Resolved merge conflict

commit 45c27e996511086632918e6d9082d1d93bc9c974
Author: Suraj Keshri <skk2142@columbia.edu>
```

```
Date:   Wed Mar 22 12:42:48 2017 -0400

    Adds parser.mly; tokens are defined

commit 3f07f48745d75a7e0a850598e5864ff979f7a8d3
Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
Date:   Wed Mar 22 00:08:30 2017 -0400

    Added script for printing scanner tokens

    For more see: https://piazza.com/class/iwrad4mnkde7gc?cid=158

commit ba3e932f9082655b6874f80275d550112438e04f
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Mar 21 23:29:47 2017 -0400

    Adds the lexer analyzer scanner.ml to .gitignore

commit f1ca37be3a9de246f81e31ca4633fe72d1ae39a0
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Mar 21 20:25:58 2017 -0400

    scanner.mll compiles successfully

commit 4897682c7cd561c5a1b288220a070f722b5419be
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Mar 21 20:20:45 2017 -0400

    Adds scannr file and test script

commit 7589855885afb4994e766b755c320b37ec8bfd6a
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Tue Mar 21 17:38:31 2017 -0400

    Updates gitignore with emacs temporary files. Adds tests directory.

commit ce146b86d499d1c63e975e22aead60bd57758b7e
Author: mesamason <mbl2109@columbia.edu>
Date:   Mon Mar 20 21:53:28 2017 -0400

    moved microC files to separate folder

commit 34fbec1b521b89d77cb168708f9733de36ba023d
Author: Suraj Keshri <skk2142@columbia.edu>
Date:   Mon Mar 20 21:15:10 2017 -0400
```

```
        added .gitginore to ignore files generated by make and running test

    commit 6cfd60403ceda7e914d48392ed3cedf60565269d
    Author: mesamason <mbl2109@columbia.edu>
    Date:   Mon Mar 20 20:56:49 2017 -0400

        clean install of microC project files

    commit 24d010a4b0b37b3db27dedea0d44a11e2c47af72
    Author: skk2142 <skk2142@columbia.edu>
    Date:   Mon Mar 20 20:40:34 2017 -0400

        Delete id_rsa.pub

    commit 10e45eb71ab0b98d4e215954f52dc7e131d973cd
    Author: Suraj Keshri <skk2142@columbia.edu>
    Date:   Mon Mar 20 20:34:42 2017 -0400

        adds a file

    commit b9114d029b732c70d774c049faa761a66966cf81
    Author: Linda Ortega <LindaOrtega@users.noreply.github.com>
    Date:   Mon Mar 20 20:26:12 2017 -0400

        Uploaded MicroC compiler example

    commit 5b88ffd01659f66cb16ae6964ec7b12641197154
    Author: Michael Lin <mbl2109@columbia.edu>
    Date:   Sun Feb 26 19:26:18 2017 -0500

        initial commit
```