Democritus

Amarto Rajaram (aar2160)
Amy Xu (xx2152)
Emily Pakulski (enp2111)
Kyle Lee (kpl2111)

“MicroC++.”

A tribute to Stephen “David” Edwards, Jr.

Goals

Provide a simple, useful language that allows a user to easily write concurrent, low-level programs. Compile to LLVM for a platform-independent IR.

1. Implement Go syntax for clarity and ease-of-use
2. Add built-in concurrency features

Syntax

- Decided to emulate Go syntax for a chance to rework the compiler front-end and emulate an interesting modern programming paradigm

```
function main() int { … }
```

- Introduced an Ocaml-style `let` keyword for variable declaration to overcome parsing errors with this syntax.

Threads & Networking

- Built-in `thread()` function allows user to create a number of threads executing a given function

```
function thread(fname string, arg *void, nthreads int)
```

- Demonstration: concurrently loading several webcomic images.
- Another built-in function uses the C Sockets API to load content from a webpage

Structs

- Robust struct implementation
  - Disallows circular struct declarations, allows nested struct implementations
  - Structs can be declared in any order regardless of dependencies

Pointers, casting, and strings

- Pointers and mallocing allows data to be passed around functions
- Malloc returns a void*, casting possible
- Even though we support pointers, we have a native `string` data type

Data Structures

- Utilizing:
  - Nested structs
  - Pointers
- LinkedList
  - Add_front
  - Add_tail
  - Delete
  - Print_list

Contributions

- **Amarto:** Threads implementation, binding C functions, sockets API, file I/O, function pointers, frontend syntax changes
- **Amy:** Structs, nested structs, pointers, pointer casting, data structure implementation
- **Emily:** language design/direction, resident Git n*zi, added some C bindings (e.g. `sleep()`, `lseek()`, `memset()`), networking + threading + file I/O contributions, frontend syntax changes.
- **Kyle:** structs, language reference manual, final project report, helping Amy with: nested structs, pointers, debugging

Lessons Learned

- **Amarto:**
  - Debugging a compiler is like playing whack-a-mole -- it's much easier to write a script to isolate the action you're trying to debug, and then gradually build it back into the compiler (thanks to David for this hint)
- **Amy:**
  - Trying to force new code to match legacy code can be more effort than it's worth. It's always okay to branch and attempt a larger rewrite if it will make everyone's lives easier. Also, be sure to understand your own syntax when writing tests.
- **Emily:**
  - Remote teamwork can be tough. Writing tests that guarantee no regressions is surprisingly difficult, especially when testing against remote files.
- **Kyle:**
  - In a team, try to play your strengths and figure out where you can help most effectively. If you think you can do something well or more efficiently than someone else, try to do it and save time - same thing works the other way (if pressed for time, let someone who knows how to do it manage it)