COMS W4115 PLT

EZAP PRESENTATION

Ryan Lee

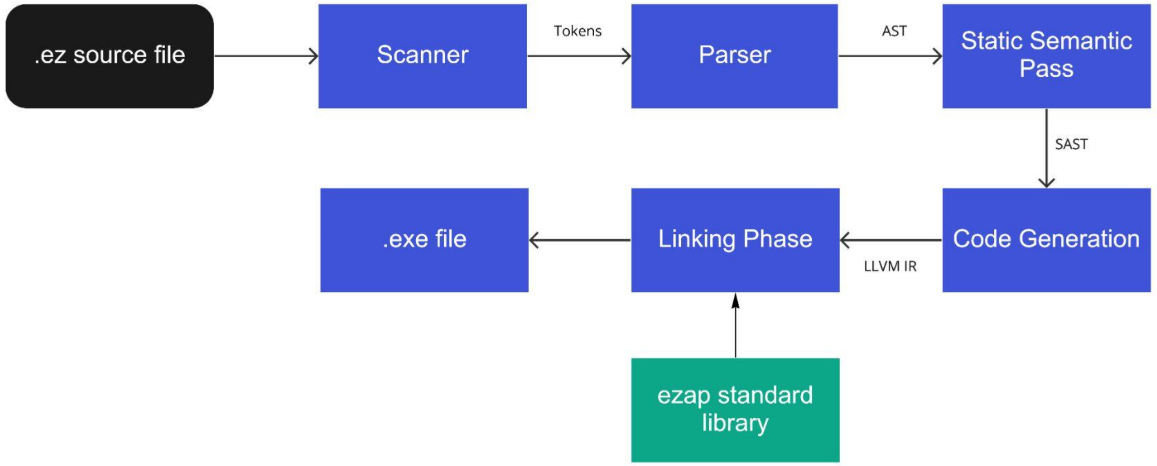
January 4th 2022

Contents

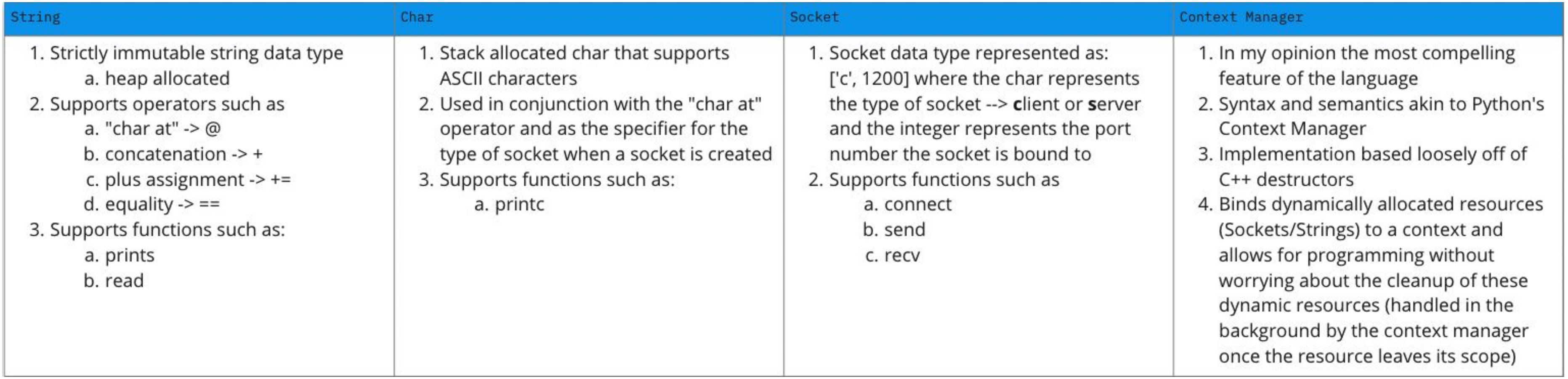
Overview

- Imperative, statically-typed language
- Superset of MicroC
- Motivated by providing intuitive features to aid in COMS 3157 programming
- C Syntax with some python/CPP features

Compiler Architecture



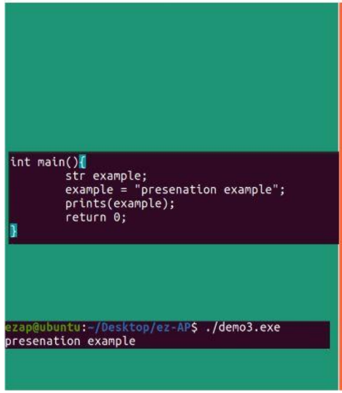
Key Language Features



String	Char	Socket	Context Manager
1. Strictly immutable string data type a. heap allocated 2. Supports operators such as a. "char at" -> @ b. concatenation -> + c. plus assignment -> += d. equality -> == 3. Supports functions such as: a. prints b. read	1. Stack allocated char that supports ASCII characters 2. Used in conjunction with the "char at" operator and as the specifier for the type of socket when a socket is created 3. Supports functions such as: a. printc	1. Socket data type represented as: ['c', 1200] where the char represents the type of socket --> **c**lient or **s**erver and the integer represents the port number the socket is bound to 2. Supports functions such as a. connect b. send c. recv	1. In my opinion the most compelling feature of the language 2. Syntax and semantics akin to Python's Context Manager 3. Implementation based loosely off of C++ destructors 4. Binds dynamically allocated resources (Sockets/Strings) to a context and allows for programming without worrying about the cleanup of these dynamic resources (handled in the background by the context manager once the resource leaves its scope)

String

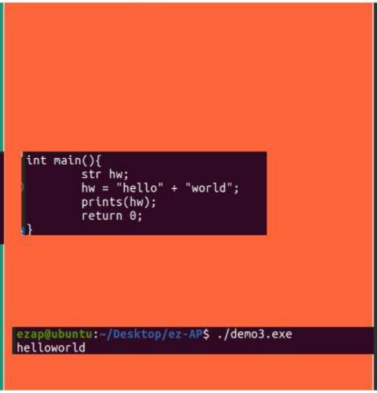
String Literal



```
int main(){
        str example;
        example = "presenation example";
        prints(example);
        return 0;
}
```

```
ezap@ubuntu:~/Desktop/ez-AP$ ./demo3.exe
presenation example
```

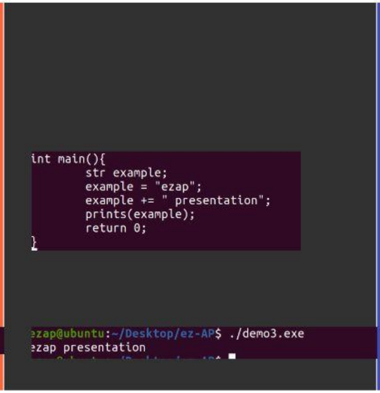
String Concatenation



```
int main(){
        str hw;
        hw = "hello" + "world";
        prints(hw);
        return 0;
}
```

```
ezap@ubuntu:~/Desktop/ez-AP$ ./demo3.exe
helloworld
```

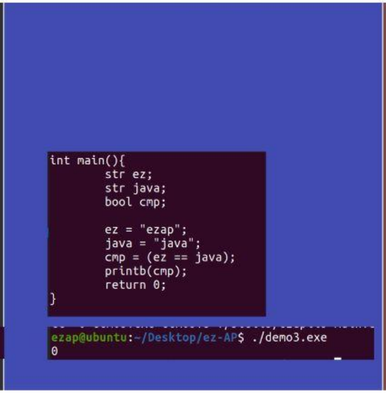
String +=



```
int main(){
        str example;
        example = "ezap";
        example += " presentation";
        prints(example);
        return 0;
}
```

```
ezap@ubuntu:~/Desktop/ez-AP$ ./demo3.exe
ezap presentation
```

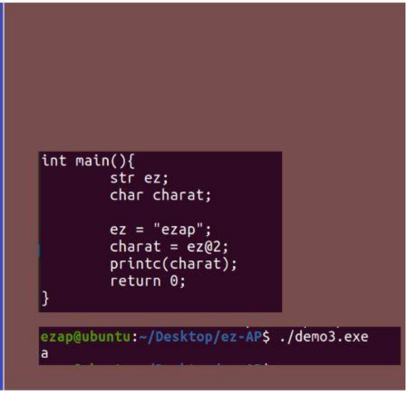
String Equality



```
int main(){
        str ez;
        str java;
        bool cmp;

        ez = "ezap";
        java = "java";
        cmp = (ez == java);
        printb(cmp);
        return 0;
}
```

```
ezap@ubuntu:~/Desktop/ez-AP$ ./demo3.exe
0
```

Char At



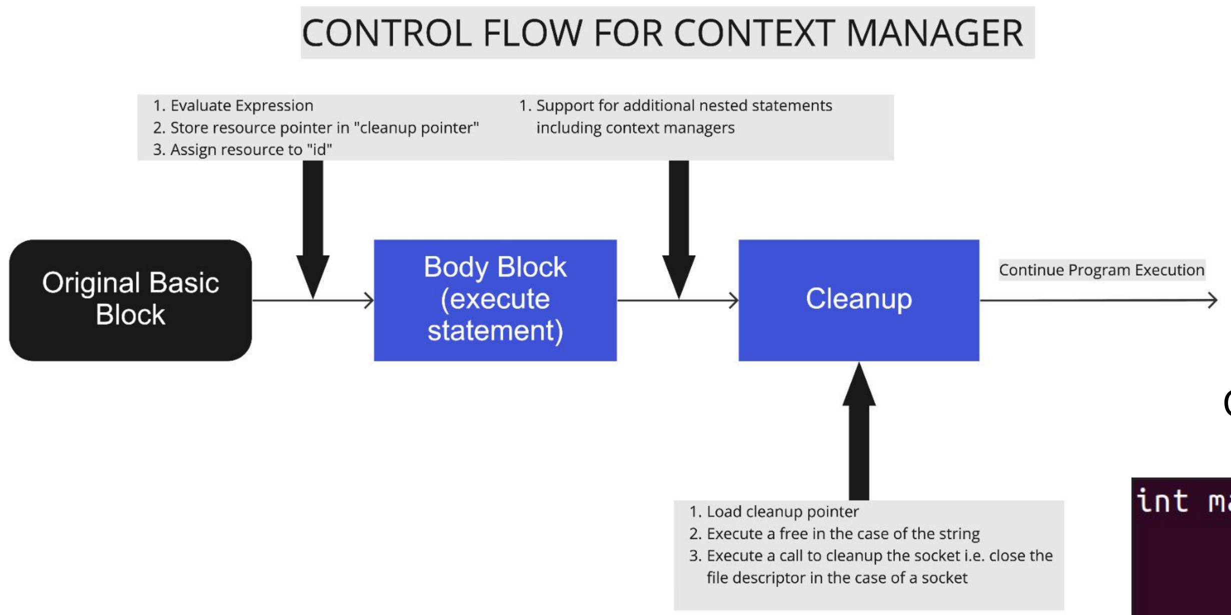
```
int main(){
        str ez;
        char charat;

        ez = "ezap";
        charat = ez@2;
        printc(charat);
        return 0;
}
```

```
ezap@ubuntu:~/Desktop/ez-AP$ ./demo3.exe
a
```

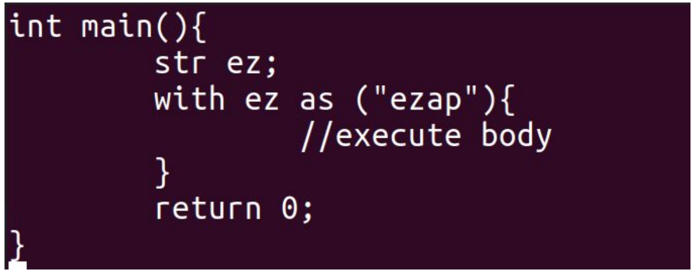
Socket

- Implemented as an LLVM struct with fields for:
  - socket type
  - port number
  - file descriptor
- At initialization the socket is a associated with a file descriptor and bound to the specified port
- Connect allows for connection to a remote host at the specified host

Context Manager


CONTROL FLOW FOR CONTEXT MANAGER
1. Evaluate Expression
2. Store resource pointer in "cleanup pointer"
3. Assign resource to "id"
1. Support for additional nested statements including context managers
Original Basic Block
Body Block (execute statement)
Cleanup
Continue Program Execution
1. Load cleanup pointer
2. Execute a free in the case of the string
3. Execute a call to cleanup the socket i.e. close the file descriptor in the case of a socket

Context Manager Syntax



```
int main(){
        str ez;
        with ez as ("ezap"){
                //execute body
        }
        return 0;
}
```

Demo

1. Chat with Netcat
2. Primitive “Web Browser”

Future Plans and Notes

- Develop server side standard library
- The language syntax shifted away from having sockets/strings appear as objects and towards maintaining C-style use of data types and function calls
- I added a requirement that non-void functions actually have a return statement that matches their declaration to avoid undefined behavior
  - void functions still do not have to have a return statement
- Special thanks to John Hui for providing me with incredibly valuable guidance to get this project off the ground and make it feasible