# 🍕 Crust

**Efficiently Process `String` using `C`**

COMS 4115 PLT Final Project Presentation

## The Team

| | |
|---|---|
| 👨‍💼 Manager | Tianqi Zhao |
| 🧑‍🔬 Language Guru | Ruiyang Hu |
| 👨‍💻 System Architect | Shaun Luo |
| 🕵️ Tester | Frank Zhang |

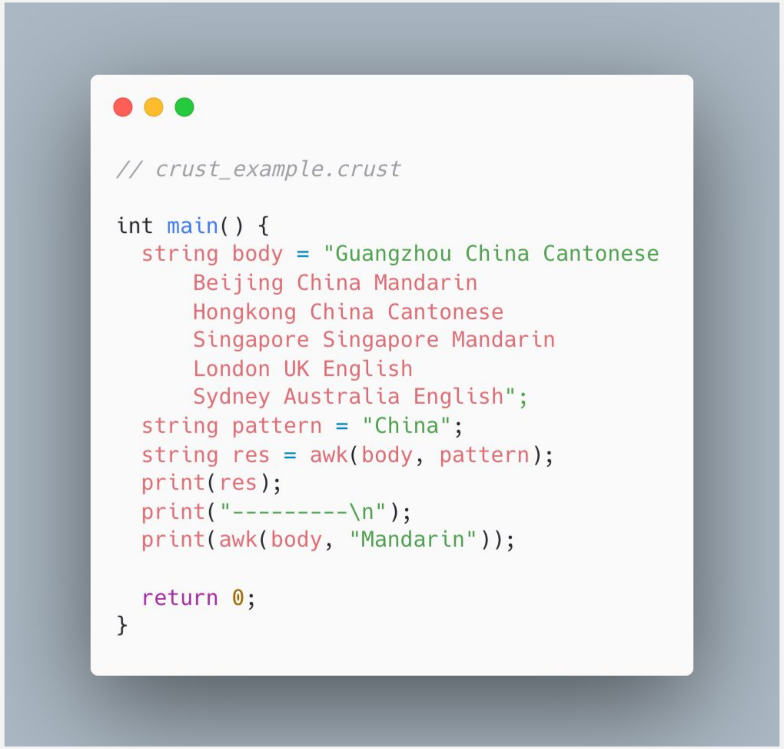
## 🍕Overview of the Project

## What is Crust?

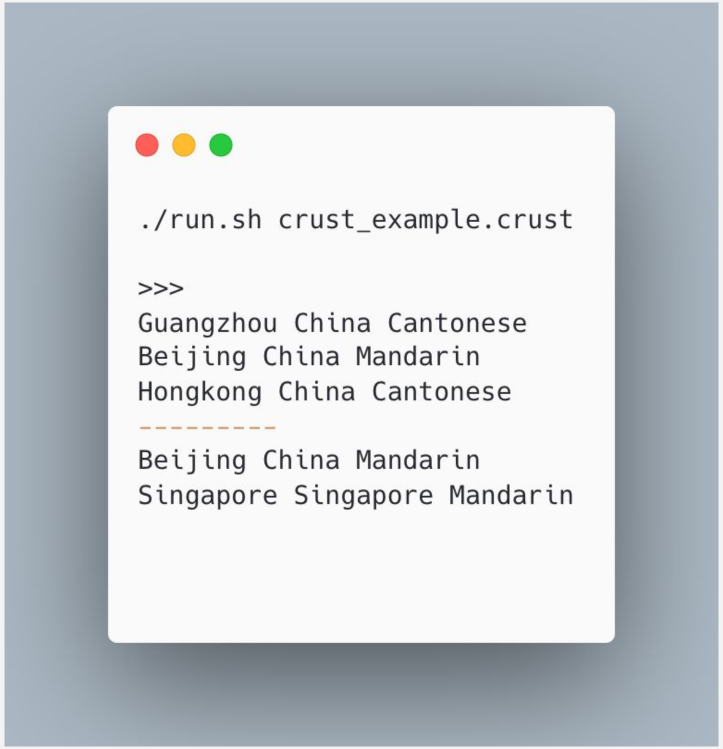
A procedural, C-like language that embeds AWK syntax and functions. It is specifically designed for efficient string processing.

The AWK Programming Language

# Show Me the Code - A Quick Example

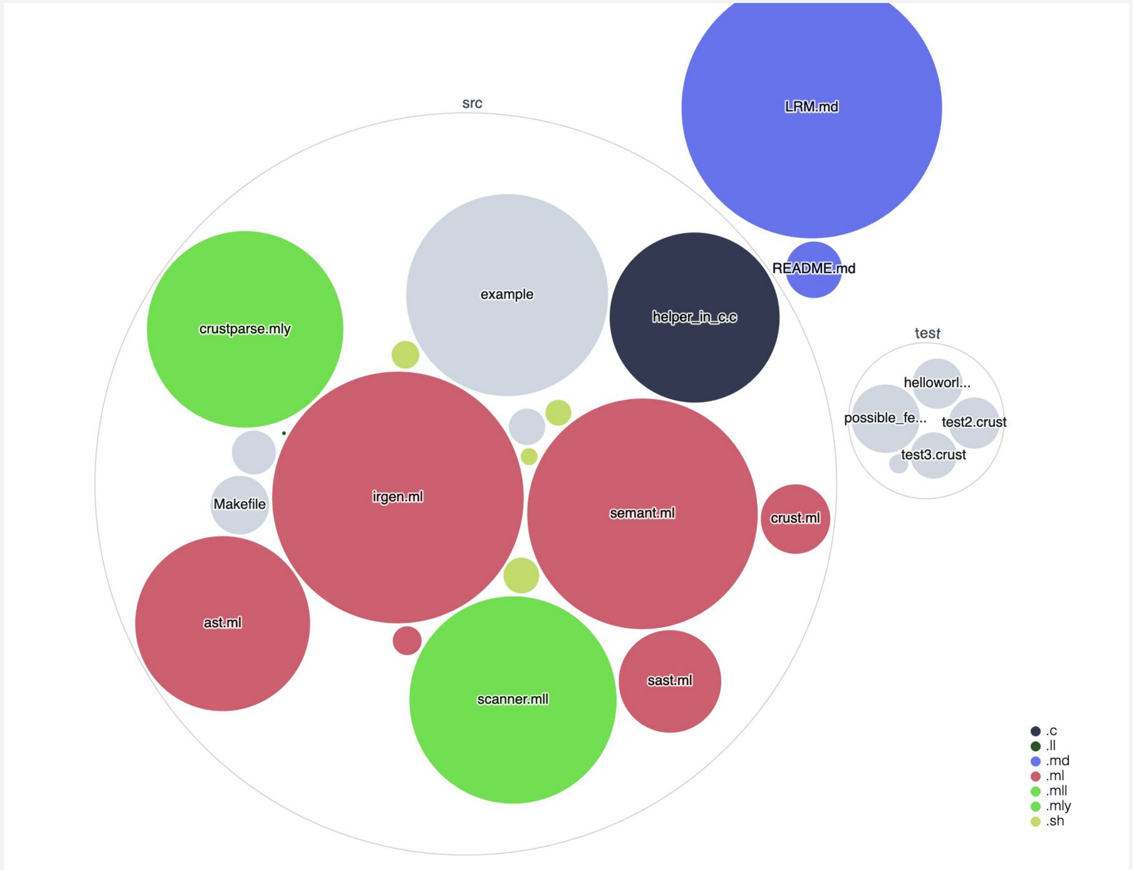


```
// crust_example.crust

int main() {
  string body = "Guangzhou China Cantonese
      Beijing China Mandarin
      Hongkong China Cantonese
      Singapore Singapore Mandarin
      London UK English
      Sydney Australia English";
  string pattern = "China";
  string res = awk(body, pattern);
  print(res);
  print("---------\n");
  print(awk(body, "Mandarin"));

  return 0;
}
```

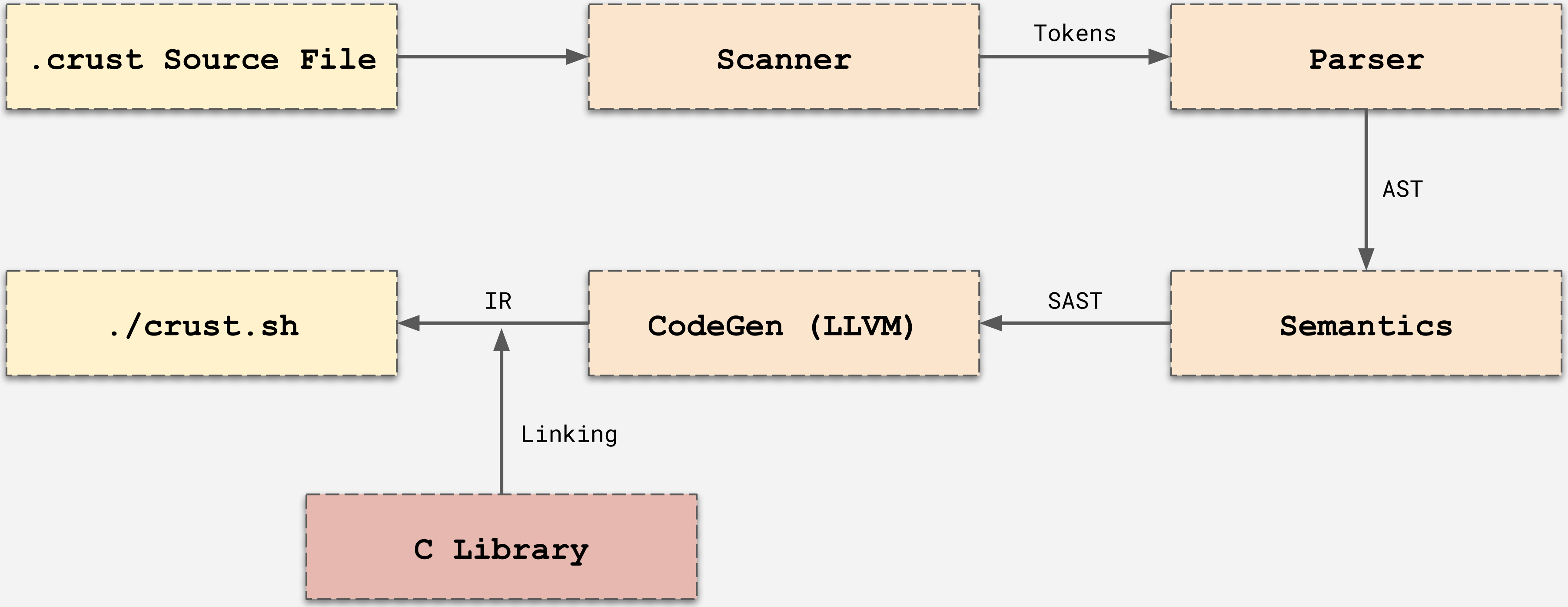


```
./run.sh crust_example.crust

>>>
Guangzhou China Cantonese
Beijing China Mandarin
Hongkong China Cantonese
---------
Beijing China Mandarin
Singapore Singapore Mandarin
```

# Code Structure



## Architectural Design



## 🍕Dive Deep into Details

## Basic Syntax

- The structure follows C-like syntax

- Data types include

  `int, float, boolean, char, string, array`

- Statements include

  `for, while, if`

## `awk` Libraries Support

```
awk

awk_line

awk_line_range

awk_line_range_start

awk_line_range_end

awk_col

…etc
```

## About `string`

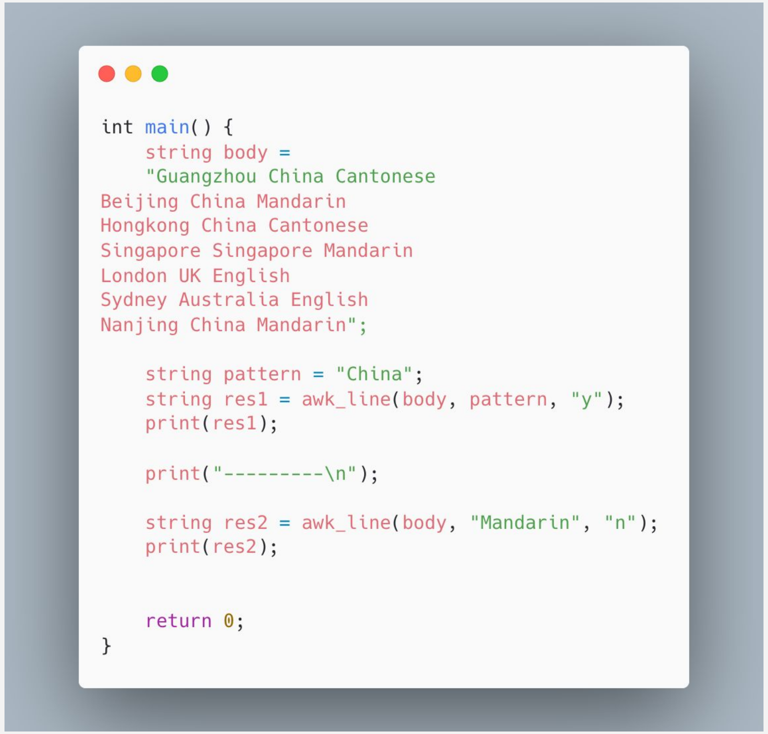
Our core features centered around string processing.

Highlights:

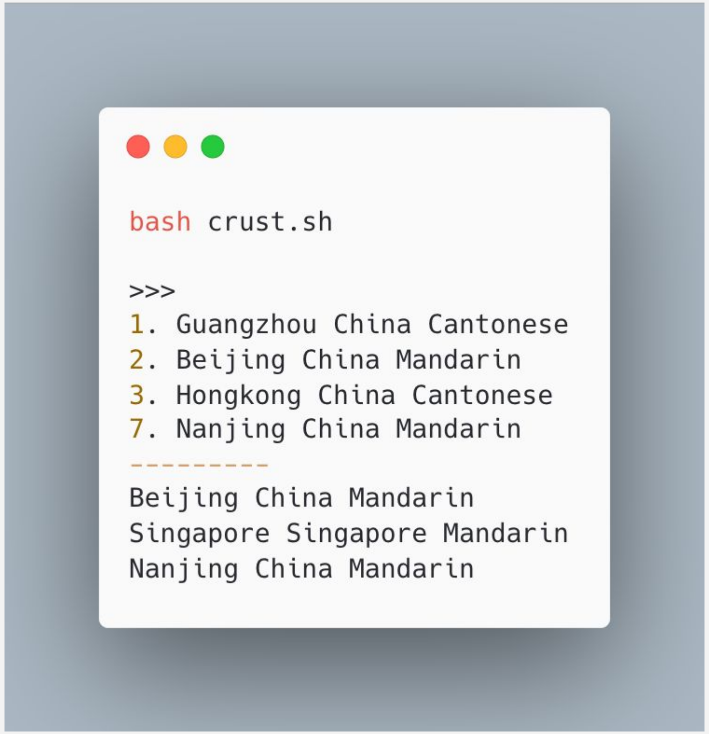
- We are able to extract escape characters, like \n, \t, etc.
- Support a lot of string processing functions:

```
str_neq
str_eq
str_concat
string_of_int
string_of_float
string_of_bool
strcmp
strlen
…etc
```

## Examples of `awk_line.crust`

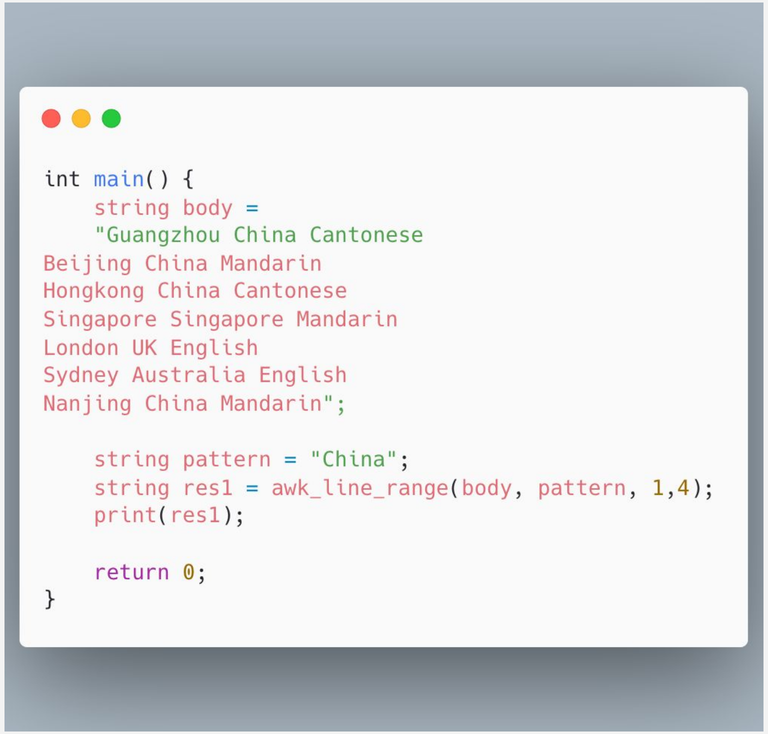


```
int main() {
    string body =
    "Guangzhou China Cantonese
Beijing China Mandarin
Hongkong China Cantonese
Singapore Singapore Mandarin
London UK English
Sydney Australia English
Nanjing China Mandarin";

    string pattern = "China";
    string res1 = awk_line(body, pattern, "y");
    print(res1);

    print("---------\n");

    string res2 = awk_line(body, "Mandarin", "n");
    print(res2);


    return 0;
}
```

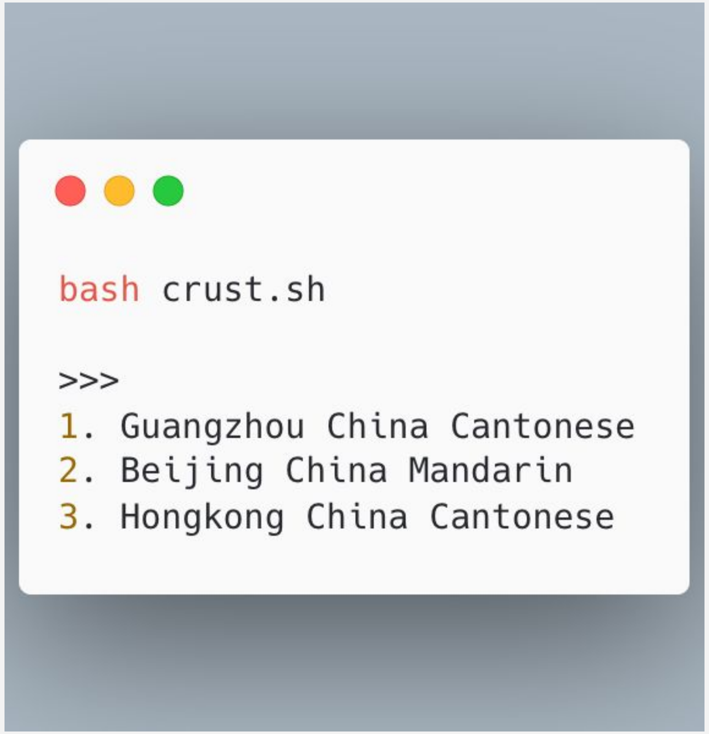


```
bash crust.sh

>>>
1. Guangzhou China Cantonese
2. Beijing China Mandarin
3. Hongkong China Cantonese
7. Nanjing China Mandarin
---------
Beijing China Mandarin
Singapore Singapore Mandarin
Nanjing China Mandarin
```

## Examples of `awk_line_range.crust`



```
int main() {
    string body =
    "Guangzhou China Cantonese
Beijing China Mandarin
Hongkong China Cantonese
Singapore Singapore Mandarin
London UK English
Sydney Australia English
Nanjing China Mandarin";

    string pattern = "China";
    string res1 = awk_line_range(body, pattern, 1,4);
    print(res1);

    return 0;
}
```



```
bash crust.sh

>>>
1. Guangzhou China Cantonese
2. Beijing China Mandarin
3. Hongkong China Cantonese
```

## Test Suite

Robust tests of Scanner, Parser, and Compilation

Tests about data types, `awk` functions, and array

Simplified command: `./crust.sh` to run all the test cases

I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO, I DO IT IN PRODUCTION
quickmeme.com

## 👾 Live Demo

## Future Roadmap

We will implement more testing on parser and scanner tonight (before graduate student deadline) to make sure there are no undetected error.

If there are more time, we would like to implement struct data type to provide more freedom on creating data types.

The last but not least, we hope to create class functions to bring more functionality to our language.