PartialC

– Mingjie Lao & Jiaying Song
Agenda

- Introduction
- Architecture
- Key Features
- Lessons Learned
- Demo
Introduction
Team Member

- Mingjie Lao: System Architect and Tester
- Jiaying Song: Manager and Language Guru
Inspiration

- Adopt data type Array to provide simple solution for dynamic programming
- Support user defined compound data types by introducing struct
What is PartialC

- Imperative language with syntax similar to C
- Lightweight and easy to implement
- Support data structures such as Array and Struct
Architecture
Compiler Architecture

file.pc → Scanner (scanner.mll) → tokens → Parser (parser.mly) → AST → Semant Check (semant.ml)

AST → SAST → Code Generation (codegen.ml)

file.pc → GCC Compiler (file.exe) → file.s → LLVM Static Compiler (file.ll)
Key Features
Key Features - General

- Primitive data types: int, float and bool
- Compound data types: string, array and struct

```
void main(){
    int int_a = -8;
    int int_b; //default value 0
    float flo_a = 2.0;
    float flo_b; //default value 0.0
    bool bool_a = true;
    bool bool_b; //default value true
}
```

```
struct Student{
    int sid;
    float grade;
    bool graduated;
};

void main(){
    string str_a = 'hello world';
    string str_b; //default is ''
    prints(str_a);

    arr int a = [1,2,3];

    Student x;
    x.sid = 1;
    x.grade = 4.5;
    x.graduated = false;
}
```
Key Features - General

Basic control flow: if if/else while for

```java
void main(){
    int a = 3;
    if (a > 1) {
        prints('correct');
    } else {
        prints('wrong');
    }
}
```

```java
void main(){
    int a = 2;
    while (a > 1){
        prints('hello world');
        a = 1;
    }
    while (a == 1){
        prints('second hello world');
        a = 2;
    }
}
```

```java
void main(){
    for (int a=1; a<5; a=a+1){
        println(a);
    }
}
```
Key Features - General

- Support user-defined function
- Support recursion

```c
void testcall(int b){
    b = b + 1;
    printf(b);
    if(b > 5){
        return 0;
    }else{
        testcall(b);
    }
}

void main(){
    int i=0;
    testcall(0);
}

void fib(arr int f, int t, float b){
    f[0] = 1;
    f[1] = 1;
    for(int i = 2; i <= t; i=i+1) {
        f[i] = f[i - 1] + f[i - 2];
    }
    printf(b);
}
```
Key Features - General

- Flexible variable declaration - NOT only at the top of function body

```c
void main(){
    int a = -8;
    printf(a);
    float b = 1.2;
    printf(b);
    bool c = true;
    bool d = false;
}
```
Key Features - Array

Array declaration in two ways: with length or with initial value

```java
void main()
{
  arr float a[8];
  a[7] = 8.0;

  arr int b = [1,2,4];
  b[1] = 9;
}
```
Key Features - Array

Array are implemented for all primitive types: int, float and bool, as well as string.

```c
void main(){
  int a = [5, 2, 1];
  float arr b[8];
  bool arr[5];
  string arr d[9];
}
```
Key Features - Array

Array out of bound check

```c
void main(){
    arr bool a[7];
    a[7] = true;
}
```

```
Root@aceweb/m妄/NAME/micro# ./single_test.sh
Fatal error: exception Failure("Array Index out of bound: 7")
```

```
./usr/lib/gcc/x86_64-linux-gnu/7/.../x86_64-linux-gnu/Sct1.o: In function `_start':
(.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
./single_test.sh: line 4: ./test.exe: No such file or directory
```
Key Features - Array

- Pass by reference when doing function call
- `sizeof` built-in function

```c
void foo(arr int t){
    t[0] = 10;
}

void main(){
    arr int a = [1,1,1,1,1];
    printi(sizeof(a));
}
```
Key Features - Struct

Support members of primitive types: int, float and bool

```c
struct Test{
    int a;
    float b;
    bool c;
};

void main(){
    Test x;
    x.a = 1;
    x.b = 4.5;
    x.c = true;

    printf(x.a);
    printf(x.b);
    if(x.c){
        printf('success!');
    }
}
```
Lessons Learned
Project never ends! START EARLY!
A simple feature implementation requires hours of debugging
Parsing rules will determine the scalability of your compiler, design wisely
Semantic check is fuzzy but really crucial at the same time
OCaml is fun!
Demo