Neat C Techniques

Sanskar Amgain

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching

Function Pointers

- Points to the address of the function
- Enables us to create first class function
- Syntax:

```
return_type (*pointer_name) (param1, param2, ....);
```

Declaration of function Pointer

Because of precedence, if we don't parenthesize the name, we declare a function returning a pointer:

```
// function returning pointer to int
int *func(int a, int b);

// function pointer that takes 2 int parameters
int (*func)(int a, int b);
```

Example

```
float function(float x) {
   return x + 2.0f;
}

int main() {
   float (*ptr) (float) = function; // ptr contains address of the function
   ptr(3.0f);
}
```

Prettifying Function Pointer

- *Typedef* creates an alias for another data type
- Using typedef can simplify the syntax of function pointer
- Syntax:

```
typedef return_type (*alias) (param1, param2, ...)
```

Typedef for function pointers

```
float function(float x) {
    return x + 2.0f;
}

typedef float (*Function_ptr)(float );

// equivalent to float (*ptr)(float )
Function_ptr ptr = function;
```

Anonymous Function

- Function that is not bound to any identifier
- Generally passed as a argument higher order function
- Created for short term use only

Anonymous function in GNU C

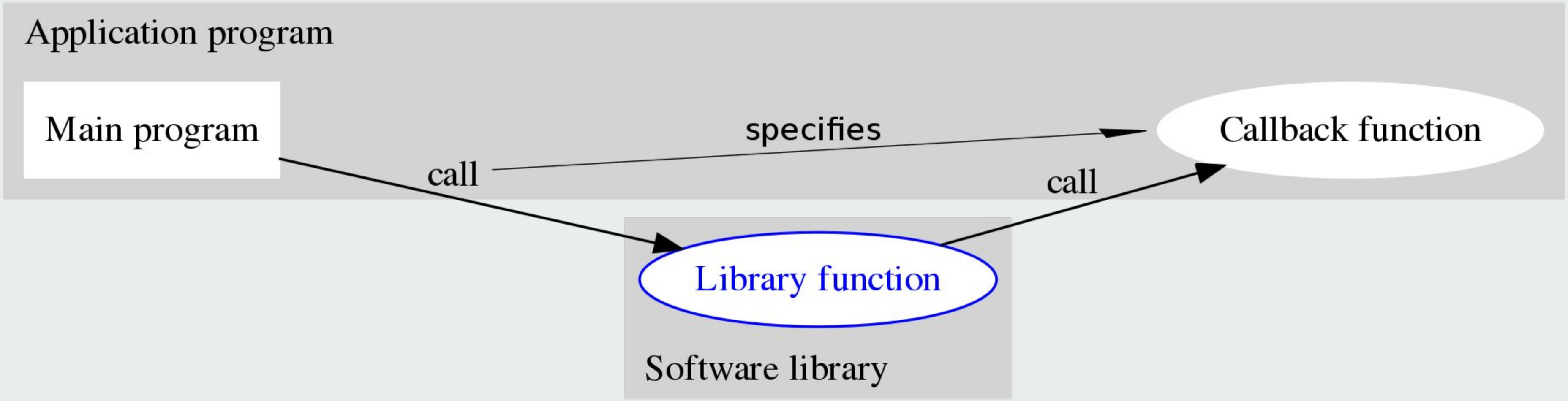
```
#define lambda(return_type, function_body) \
  ({ \
    return_type anon_func_name_ function_body \
    anon_func_name_; \
  })


int (*max)(int, int) = lambda (int, (int x, int y) { return x > y ? x : y; });
```

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching

Callback Function

- Function that is passed to another function as argument
- Callback is generally called when some kind of event occurs.
- Callback in C can be implemented by passing a function pointer.


Application program
Main program
specifies
Callback function
call
call
Library function
Software library

Example: *callback_newton_raphson*

Example: *map*

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching
- Coroutines

Polymorphism

- Polymorphism is the use of single symbol to represent multiple types.
- In C polymorphism can be implemented using ***void*** *.
- ***void*** * can point to data of any type.

Example: *bubble_sort*

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching

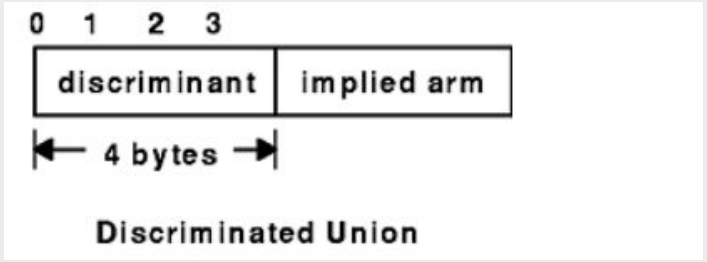
Union

- Struct with all elements allocated at the same address
- Naturally, union can hold element for only for one member at a time
- Size of the union is the size of the largest member

What if elements of union have different types?

Discriminated Union

- Also called tagged union
- Discriminated union contains a the type of data that is stored in union
- Contains both discriminant and a component
- Component types are called *arms* of the union



Example: *Simple Parser*

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching
- Coroutines

How else do we implement polymorphism in C?

Example: *Switched Case*

Example: *Function Dispatch*

Content

- Function Pointers
- Callback Functions
- Polymorphism with Void Pointer
- Discriminated Unions
- Switch vs Function Dispatching
- Coroutines

Coroutines

- Coroutines are functions that can be suspended and resumed.
- Coroutine holds state between invocation of routines.

References

Introduction To Function Pointer

Discriminated Union IBM

Introduction to *typedef*

Anonymous function in C

Discriminated Union (Medium)