Day08 A

Young W. Lim

2017-10-14 Sat

Outline

1. Based on

2. C Functions (2) Storage Class and Scope
   - Storage Class Specifiers
   - A. Storage Duration
   - B. Scope
   - C. Linkage

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

Storage Class

- the storage class specifier
  - `auto`
  - `register`
  - `extern`
  - `static`

- an identifier's storage class and scope rules *determine*
  - storage duration
  - scope
  - linkage

Identifier of interests

- an identifier
  - a variable name
  - a function name

Identifier Attributes

- storage duration (temporal)
  - the period during which an identifier exists in memory
- scope (spatial)
  - where an identifier can be referenced in a program
- linkage
  - determines whether an identifier is known only in the current file or in any other file

Classification of Identifier Attributes

storage duration
automatic storage duration
static storage duration

scope
function scope
file scope *
block scope *
function prototype scope

linkage
external linkage
internal linkage

Storage duration

- storage duration
  - the period during which an identifier exists in memory
  - some exists briefly and are repeatedly created and destroyed (variables defined inside a function)
  - others exists for the program's entire execution (variables defined outside all functions)
  - automatic storage duration (`auto` + scope)
  - static storage duration (`static` + scope)

Automatic Storage Duration

- automatic storage duration variables are created
  - when the program control is entered a block `{...}`
  - where the variable was defined
- automatic storage duration variables exists
  - while the block is active
  - (while the control is in the block)

Static Storage Duration

- `extern` and `static` in the declaration of variables and functions
- static (storage) variables and static (storage) functions exist from the program starts and until the program ends
- static (storage) variables are allocated and initialized only once before the program executes

- `extern` / `static` global variables
- `extern` / `static` functions
- `static` local variables

Storage Class and Linkage

	functions	global variables	local variables
extern	static storage external linkage	static storage external linkage	static storage *
static	static storage internal linkage	static storage internal linkage	auto storage NA

Local Variables

- only variables, not a function can have automatic storage duration
- all functions and global variables have static storage duration
- functions's local variable
  - `func(int a) { int b, c; .... }`
  - these variables have automatic storage duration by default
  - `func(auto int a) { auto int b, c; .... }`
  - refered as automatic variables

Global Variables

- global variables
  variables declared outside all function definitions
- functions
  are declared always outside any function definition

- global variables and functions
  - `extern` storage class by default
  - any functions can reference global variables and functions
  - these functions must be defined / declared
    after global variables and functions in the file

Static Local Variables

- local variables with `static` keyword
- known only in the function
  where the local variables are defined
- retain the values when the function exits
  (the value is preserved across function calls)
- can start with the retained value
  when the function is called again
- initialized with zero by default
  when no explicit initialization exists

Scope

- scope
  - where an identifier can be referenced in a program

- some can be referenced throughout a program (global variable)
- others from only portions of a program (local variable)

Scope Types

- function scope ..... `func(...) { ... }`
- file scope ..... `t1.c t2.c`
- block scope ..... `{ ... }`
- funciton prototype scope .... `func(...);`

Function Scope

- labels are the only identifiers with function scope
  - start*:*
  - `goto`
  - `switch`
- labels can be used anywhere in the function
- labels cannot be referenced outside the function body

File Scope

- identifiers declared outside any function
- known (accessible) in all functions
  from the point at which the identifier is declared
  until the end of the file

- gloabl variables
- function definitions
- function prototypes

Block Scope

- identifiers defined inside a block
- block scope ends at the terminating right brace (}) of the block
- local variables defined at the beginning of a function
- function parameter variables also have block scope
- any block can have its own variable definitions
- static local variable still have block scope
  but static storage duration

Hiding Block Scope

- blocks can be nested
  - identifiers of the outer block and inner block can have the same name
  - then the outer identifier is hidden by the inner identifier (higher priority)

Function Prototype Scope

- the only identifer is the parameter list of a function prototype
- function prototypes require
  - no variable names in the parameter list
  - only types in the parameter list
- the compiler ignores the variable name
- the identifers used in a functon prototype
  can be reused elsewhere in the program

Linkage

- linkage
  - determines for a multiple-source-file program
    whether an identifier is known
    only in the current source file or
    in any source file with proper declarations

- `static` prevents an identifier from being referenced in other files
  - `static` global variables
  - `static` funtions

- `extern` indicates an identifer is defined
  either later in the same file or in a different file
  - `extern` global variables
  - `extern` funtions

Internal Linkage

- it is possible to restrict the scope of a variable or a function to the file in which it is defined
- can be prevented from being used by any function that are defined in other files

- static gloabal variables
- static functions

External Linkage

- non-static gloabal variables
- non-static functions
- they can be accesed in other files
  if those files contain proper declaration and/or function prototypes