Day08 A

Young W. Lim

2017-10-28 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)
  - the portion of a program where an identifier can be referenced
- 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 Examples

```
#include <stdio.h>

void func() {
  int i = 0;

  printf("i= %d\n", i);  i++;
}

int main(void) {
  func();
  func();
  func();
  func();
}
---
i= 0;
i= 0;
i= 0;
i= 0;
```

- automatic storage duration

```
#include <stdio.h>

void func() {
  static int i = 0;

  printf("i= %d\n", i); i++;
}

int main(void) {
  func();
  func();
  func();
  func();
}
---
i= 0;
i= 1;
i= 2;
i= 3;
```

- static storage duration

Scope Examples

```
#include <stdio.h>

void func() {
  int i = 111;

  printf("1.i= %d\n", i);

  { int j = 222;

    printf("2.i= %d\n", i);
    printf("2.j= %d\n", j);

    { int k = 333;

      printf("3.i= %d\n", i);
      printf("3.j= %d\n", j);
      printf("3.k= %d\n", k);
    }

    printf("2.i= %d\n", i);
    printf("2.j= %d\n", j);
  }

  printf("1.i= %d\n", i);

}

int main(void) {

  func();

}
```

```
---
1.i= 111
2.i= 111
2.j= 222
3.i= 111
3.j= 222
3.k= 333
2.i= 111
2.j= 222
1.i= 111
```

- block scope

Linkage Examples

```
// t1.c --------------------
#include <stdio.h>

void func1(void) {
  puts("func1 is called");
}

void func2(void) ;
void func3(void) ;

int main(void) {

   func1();
   // func2();
   func3();

}
```

- func2 cannot be called in main
- func3 can be called in main

```
// t2.c --------------------
#include <stdio.h>

static
void func2(void) {
  puts("func2 is called");
}

void func3(void) {
  printf("func3: ");
  func2();
}

---
gcc -c t1.c
gcc -c t2.c
gcc t1.o t2.o
```

- internal linkage : func2
- external linkage : func3

Storage duration

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

Automatic Storage Duration

automatic storage duration variables

- defined inside a block
- created (allocated) whenever the program control is entered the block {...}
- exists while the block is active
  (while the control is in the block)
- destroyed (dealocated) whenenver the program control is exited from the block {...}

Static Storage Duration

static storage duration variables / functions

- defined by using either specifier
  - `extern`
  - `static`
- exist
  - from the program starts and
  - until the program ends

static storage duration variables

- allocated and initialized only once
  before the program executes

Static Storage Duration Variables and Functions

- global variables with `extern` keyword
- global variables with `static` keyword
- local variables with `extern` keyword
- local variables with `static` keyword
- functions with `extern` keyword
- functions with `static` keyword

Storage Class and Linkage

- with `extern` keyword

extern	functions	global variables	local variables
duration	static storage	static storage	static duration
linkage	external linkage	external linkage	external linkage

- with `static` keyword

static	functions	global variables	local variables
duration	static storage	static storage	static storage
linkage	internal linkage	internal linkage	NA

Local Variables

- all functions and global variables defined outside a function
  - `extern` by default
  - static storage duration
- only variables, not a function
  - automatic storage duration
- variables defined in a function
  - `auto` by default
  - `func(int a) { int b, c; .... }`
  - `func( auto int a) { auto int b, c; .... }`
  - refered as automatic variables

Global Variables and Functions

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

- global variables and functions can be defined with
  - `extern` storage class
  - `static` storage class

Referencing Global Variables and Functions

global variables and functions with

- `extern` storage class (_by default_)
  - any functions can reference external linkage variables / functions
- `static` storage class
  - only functions in the same file can reference internal linkage variables / functions
  - all these referencing functions must be defined / declared after the referenced 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 once by default
  when no explicit initialization exists

Scope

- scope
  - the portion of a program where a given identifier can be referenced

- 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 start;

  - switch (exp) {
  - case label : (integer label)
  - default :
  - }
- labels can be used anywhere in the function
- labels cannot be referenced outside the function body

File Scope

- identifiers declared / defined outside any function
  - gloabl variables
  - function definitions
  - function prototypes
- known (accessible) in all functions which are defined / declared
  - from the point at which the identifier is declared
  - to the end of the file

Block Scope

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

Hiding Block Scope

- blocks can be nested
  - identifiers of the outer block
  - identifiers of the 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 of a function prototype is the parameter variable list
- function prototypes require
  - no variable names
  - only types
  - in the parameter list
- the variable name of a function prototype
  - is ignored by the compiler
  - 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 other source file with proper declarations

Linkage Qualifiers

- `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

- to restrict the scope of a variable or a function
  to the file in which it is defined
- to prevent from being referenced by any function
  that are defined in other files

- static gloabal variables
- static functions

External Linkage

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