Modularity in C

Giovanni Squillero
Politecnico di Torino - DAUIN
giovanni.squillero@polito.it

Scope

- An identifier is *visible* (i.e., it can be used) only within a region of program text called its *scope*
- Four kinds of scopes
  - function
  - file
  - block
  - function prototype.

Scope

- An identifier is *visible* (i.e., it can be used) only within a region of program text called its *scope*
- Four kinds of scopes
  - function
  - file
  - block
  - function prototype

only labels!

within a prototype

Scope: Example

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
   int lcl1;
   …
}
int myFunction2(int parameter)
{
   int lcl2;
   …
}
```


Scope: Example

FILE

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
   int lcl1;
   …
}
int myFunction2(int parameter)
{
   int lcl2;
   …
}
```


Scope: Example

BLOCK

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
   int lcl1;
   …
}
int myFunction2(int parameter)
{
   int lcl2;
   …
}
```

Storage duration

- An object has a storage duration that determines its *lifetime*.
- There are three storage durations:
  - static
  - automatic
  - allocated

Storage duration

- An object has a storage duration that determines its *lifetime*.
- There are three storage durations:
  - static
  - automatic
  - ~~allocated~~

Next lesson

Initialization

- All objects with static storage duration are *initialized* (set to their initial values) before program startup
- Automatic objects are not initialized by default

Storage classes specifiers

- typedef
- extern
- static
- auto
- register

Storage classes specifier

"syntactic convenience only"

- ~~typedef~~
- extern
- static
- auto
- register

Storage classes specifier

AUTOMATIC STORAGE DURATION

- typedef
- extern
- static
- auto
- register

Storage classes specifier

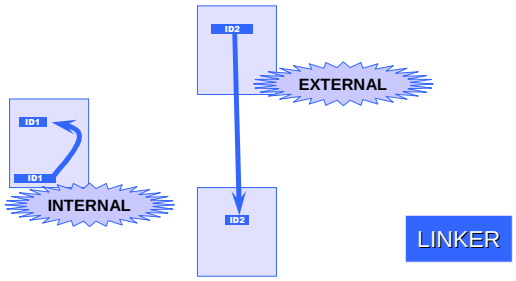
- typedef
- extern
- static
- auto
- register

STATIC STORAGE DURATION

Linkages of identifiers

- External linkage
- Internal linkage
- No linkage

Linkages of identifiers



Linkages: Example

```
#include <stdio.h>

int Glob;
static int _Private;

int myFunction(int parameter)
{
  int lcl;
  extern ref;

  …

  return 0;
}
```


Linkages: Example

NO LINKAGE

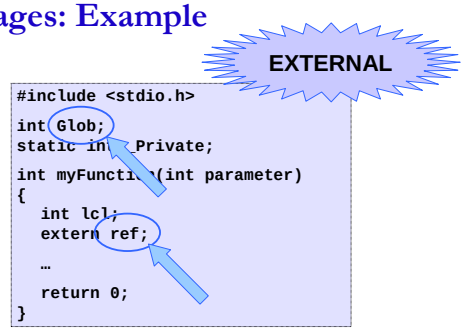
```
#include <stdio.h>

int Glob;
static int _Private;

int myFunction(int parameter)
{
  int lcl;
  extern ref;

  …

  return 0;
}
```


Linkages: Example

INTERNAL

```
#include <stdio.h>

int Glob;
static int _Private;

int myFunction(int parameter)
{
  int lcl;
  extern ref;

  …

  return 0;
}
```

Linkages: Example



Summary

- Scope
  - Where an identifier can be used
- Storage
  - When an object can be used
- Linkage
  - When an object can be used

What about functions?

- Function identifiers
  - Always file scope
  - Always static storage duration
  - Either external (default) or internal (*static* functions) linkage