

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 *only labels!*
 - file
 - block
 - function prototype *within a prototype*

Scope: Example

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
    int lc11;
    ...
}
int myFunction2(int parameter)
{
    int lc12;
    ...
}
```

Scope: Example

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
    int lc11;
    ...
}
int myFunction2(int parameter)
{
    int lc12;
    ...
}
```

FILE

Scope: Example

```
#include <stdio.h>
int Glob;
int myFunction(int parameter)
{
    int lc11;
    ...
}
int myFunction2(int parameter)
{
    int lc12;
    ...
}
```

BLOCK

Storage duration

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

G. Squallero Advanced Programming 7

Storage duration

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

Next lesson

G. Squallero Advanced Programming 8

Initialization

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

G. Squallero Advanced Programming 9

Storage classes specifiers

- typedef
- extern
- static
- auto
- register

G. Squallero Advanced Programming 10

Storage classes specifier

- typedef "syntactic convenience only"
- extern
- static
- auto
- register

G. Squallero Advanced Programming 11

Storage classes specifier

- typedef
- extern
- static
- auto
- register

AUTOMATIC STORAGE DURATION

G. Squallero Advanced Programming 12

Storage classes specifier

- typedef
- extern
- static
- auto
- register

STATIC STORAGE DURATION

G. Squallero Advanced Programming 13

Linkages of identifiers

- External linkage
- Internal linkage
- No linkage

G. Squallero Advanced Programming 14

Linkages of identifiers

The diagram illustrates the linker's role in resolving identifiers. It shows two boxes representing source files. The left box contains 'ID1' with a blue arrow pointing to a starburst labeled 'INTERNAL'. The right box contains 'ID2' with a blue arrow pointing to a starburst labeled 'EXTERNAL'. A blue box labeled 'LINKER' is positioned between them, with a blue arrow pointing from the 'INTERNAL' starburst to the 'EXTERNAL' starburst, indicating the linker's process of resolving the external reference.

G. Squallero Advanced Programming 15

Linkages: Example

```
#include <stdio.h>
int Glob;
static int _Private;
int myFunction(int parameter)
{
    int lcl;
    extern ref;
    ...
    return 0;
}
```

G. Squallero Advanced Programming 16

Linkages: Example

NO LINKAGE

```
#include <stdio.h>
int Glob;
static int _Private;
int myFunction(int parameter)
{
    int lcl;
    extern ref;
    ...
    return 0;
}
```

The code is identical to slide 16. A blue starburst labeled 'NO LINKAGE' is placed above the code. Blue arrows point from the starburst to the 'extern ref;' line and the 'parameter' argument in the function signature, indicating that these elements do not have linkage.

G. Squallero Advanced Programming 17

Linkages: Example

INTERNAL

```
#include <stdio.h>
int Glob;
static int _Private;
int myFunction(int parameter)
{
    int lcl;
    extern ref;
    ...
    return 0;
}
```

The code is identical to slide 16. A blue starburst labeled 'INTERNAL' is placed above the code. A blue arrow points from the starburst to the 'static int _Private;' line, indicating that this variable has internal linkage.

G. Squallero Advanced Programming 18

Linkages: Example

```
#include <stdio.h>
int Glob;
static int Private;
int myFunction (int parameter)
{
    int lcl;
    extern ref;
    ...
    return 0;
}
```

EXTERNAL

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