Princeton University
COS 217: Introduction to Programming Systems
C Variable Declarations and Definitions

Variable **declaration** is a statement that informs the compiler of the name, type, scope, linkage, and duration of the variable. A variable **definition** is a declaration that causes the compiler to allocate memory.

Scope (compiletime concept)

file: The variable is accessible within the file in which it is declared, from the point of declaration to the end of the file.

block: The variable is accessible within the block in which it is declared, from the point of declaration to the end of the block.

Linkage (linktime concept)

external: The variable is accessible from multiple files.

internal: The variable is accessible from only the file in which it is declared.

Duration (runtime concept)

temporary: The variable exists only during the execution of the function or block in which it is declared. Physically, the variable's value is stored in the runtime Stack.

process: The variable exists throughout the entire process. Physically, the variable's value is stored in the Data Section (if the programmer specifies an initial value) or the BSS Section (if the programmer does not specify an initial value). The variable's value is initialized at program startup. If in the BSS section, its initial value is 0.

C Code	Decl/Def	Scope	Linkage	Duration	Comment
`int a = 5;`	definition	file	external	process	
`int b;`	declaration*	file	external	process	
`extern int c = 5;`	definition	file	external	process	
`extern int d;`	declaration	file	external	process	
`static int e = 5;`	definition	file	internal	process	
`static int f;`	definition	file	internal	process	
`void fun(int g) {`	definition	block	internal	temporary	Common
`  int h = 5;`	definition	block	internal	temporary	Common
`  int i;`	definition	block	internal	temporary	Common
`  extern int j = 5;`	ILLEGAL				
`  extern int k;`	declaration	block	UNKNOWN	process	Rare
`  static int l = 5;`	definition	block	internal	process	
`  static int m;`	definition	block	internal	process	
`  ...`					
`}`					

* Special rule: If no definition appears in any other .c file, this becomes a definition.

Examples of Global Variable Declarations and Definitions

Suppose a program consists of file1.c and file2.c (only). Consider these combinations of global variable declarations and definitions:

Example #	file1.c	file2.c	Result
1	`int i = 5;`	`int i = 5;`	`Error. Multiple def of i.`
2	`int i = 5;`	`int i;`	`OK.`
3	`int i = 5;`	`extern int i = 5`	`Error. Multiple def of i.`
4	**`int i = 5;`**	**`extern int i;`**	**`OK. Normal.`**
5	`int i = 5;`	`static int i = 5;`	`OK.`
6	`int i = 5;`	`static int i;`	`OK.`
7	`int i;`	`int i;`	`OK. Relies on special rule.`
8	`int i;`	`extern int i = 5;`	`OK.`
9	`int i;`	`extern int i;`	`OK. Relies on special rule.`
10	`int i;`	`static int i = 5;`	`OK. Relies on special rule.`
11	`int i;`	`static int i;`	`OK. Relies on special rule.`
12	`extern int i = 5;`	`extern int i = 5;`	`Error. Multiple def of i`
13	`extern int i = 5;`	`extern int i;`	`OK.`
14	`extern int i = 5;`	`static int i = 5;`	`OK.`
15	`extern int i = 5;`	`static int i;`	`OK`
16	`extern int i;`	`extern int i;`	`Error. No def of i.`
17	`extern int i;`	`static int i = 5;`	`Error. No def of i.`
18	`extern int i;`	`static int i;`	`Error. No def of i.`
19	**`static int i = 5;`**	**`static int i = 5;`**	**`OK. Normal`**
20	**`static int i = 5;`**	**`static int i;`**	**`OK. Normal.`**
21	**`static int i;`**	**`static int i;`**	**`OK. Normal.`**