**Review: Four** areas of memory that your code works with:

* The **static** data area contains a **code** section (typically read-only) and a section for **globals**
* The **stack** is used by the system for runtime information and for local variables
* The **freestore (heap)** is used for programmer-defined storage

Review: Scope, storage, duration

* Multi-file programs have to place their variables in the correct “block” and then communicate
* **Scope:** where a name is visible
  + **block, file**
* **Duration:** how long an object stays in memory
  + **Static**, **automatic**, **programmer-defined**
* **Linkage**: how a name is shared with different units
  + **None**, **external**, **internal**
* There are two keywords that control these items:
  + **Extern** to change the linkage of global variables
  + **Static** to change the scope or duration of an object

Pointers: Two big ideas coming up

* Creating and using **pointer** variables
  + Declaring pointers to different types
  + Using the address-of operator
  + Using the indirection operator
* Pointer arithmetic
  + Using pointers with array types

Variable review

* All variables have three attributes
  + Name: used instead of memory address
  + Type: determines what can be stored in the variable and the valid operations
  + Value: the data or state stored

**TWO NEW OPERATORS**

* The **address-of** operator (**&**) returns the address of a variable in memory
  + Will normally be printed as a hex integer
* The **sizeof** operator returns the size of a variable or type in bytes

**Declaring and Defining Pointers**

* Create a pointer like this:
  + **typeOfPointee \* nameOfPointer;**
  + **Int \* iPtr;**
  + read(r to left) as: iPtr is a pointer to int
  + Space before or after \* is immaterial
  + Every pointer needs its own \*
  + **Int \* iPtr, i;** //iPtr is a pointer; i is an int

The Indirect Value

* The value the pointer “points to” is retrieved by the **indirection operator**
  + Unary operator (**\***) applied to a pointer
  + (good slide w/ a bunch of examples)
* 