Chapter 3

(3.2, 3.5)

Informal Semantics:
Names, Bindings, and Lifetime

Lifetime

Time during execution that an object remains bound to memory space

- *Static*
  - Allocated for entire program execution
- *Automatic*
  - Allocated at block entry & deallocated at exit
- *Dynamic*
  - Allocated & deallocated as needed by program during execution

Categories of Variables by Lifetimes

Static allocation for

- code
- globals
- static variables
- explicit constants (strings, sets, etc.)

Categories of Variables by Lifetimes

Stack-dynamic -- Storage bindings are created for variables when their declaration statements are executed

Advantage: allows recursion; conserves storage

Disadvantages:

- Overhead of allocation and deallocation
- Inefficient references (indirect addressing)

Lifetime and Storage Management

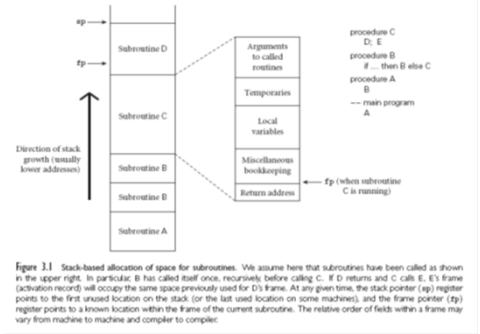

Figure 3.1 Stack-based allocation of space for subroutines. We assume here that subroutines have been called as shown in the upper right. In particular, B has called itself once, recursively, before calling C. If D returns and C calls E, E's frame (activation record) will occupy the same space previously used for D's frame. At any given time, the stack pointer (sp) register points to the first unused location on the stack (or the last used location on some machines), and the frame pointer (fp) register points to a known location within the frame of the current subroutine. The relative order of fields within a frame may vary from machine to machine and compiler to compiler.

Categories of Variables by Lifetimes

Heap-dynamic -- Allocated and deallocated by explicit directives, specified by the programmer, which take effect during execution

Referenced only through pointers or references, e.g. dynamic objects in C++ (via `new` and `delete`), all objects in Java

Advantage: provides for dynamic storage management

Disadvantage: inefficient and unreliable

Lifetime and Storage Management

Heap for dynamic allocation

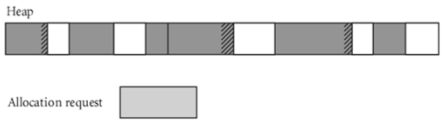

Figure 3.2 Fragmentation. The shaded blocks are in use; the clear blocks are free. Cross-hatched space at the ends of in-use blocks represent internal fragmentation. The discontiguous free blocks indicate external fragmentation. While there is more than enough total free space remaining to satisfy an allocation request of the illustrated size, no single remaining block is large enough.

Persistence

The lifetime of program objects are determined by the execution time of the program

The lifetime of data often extends beyond program execution

Objects are *persistent* if they continue to exist between executions of programs

Current languages use files for this, but research is being done on persistent programming languages

The Meaning of Names within a Scope

Aliasing

- If two variable names can be used to access the same memory location, they are called aliases
- Aliases are created by pointers, reference variables, C and C++ unions, parameter passing
- Aliases are harmful to readability

The Meaning of Names within a Scope

Overloading

Same name used for different subprograms

- The signatures of the subprograms are used to distinguish between them

Some overloading happens in almost all languages

- integer + and real +
- println in Java