Previous: 14.1.7 Storage Class for Functions
Up: 14.1 Storage Classes
Previous Page: 14.1.7 Storage Class for Functions
Next Page: 14.2 Dynamic Memory Allocation

14.1.8 Stack vs Heap Allocation

We conclude our discussion of storage class and scope by breifly describing how the memory of the computer is organized for a running program. When a program is loaded into memory, it is organized into three areas of memory, called *segments*: the *text segment*, *stack segment*, and *heap segment*. The text segment (sometimes also called the code segment) is where the compiled code of the program itself resides. This is the machine language representation of the program steps to be carried out, including all functions making up the program, both user defined and system.

The remaining two areas of system memory is where storage may be allocated by the compiler for data storage. The stack is where memory is allocated for automatic variables within functions. A stack is a *Last In First Out* (LIFO) storage device where new storage is allocated and deallocated at only one ``end'', called the Top of the stack. This can be seen in Figure 14.13.

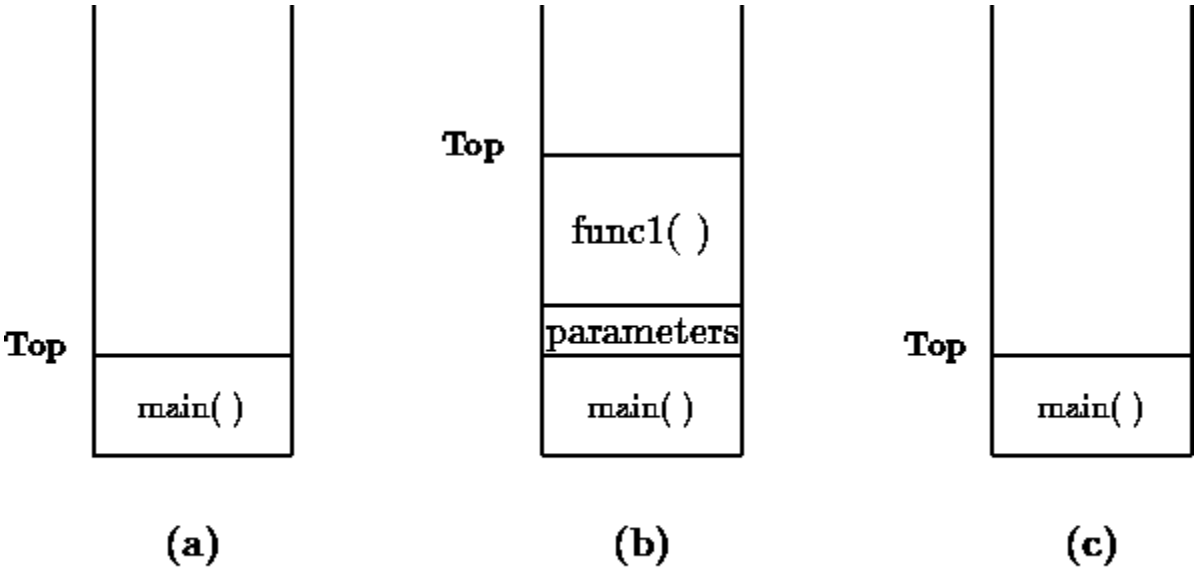

Figure 14.13: Organization of the Stack

When a program begins executing in the function `main()`, space is allocated on the stack for all variables declared within `main()`, as seen in Figure 14.13(a). If `main()` calls a function, `func1()`, additional storage is allocated for the variables in

`func1()` at the top of the stack as shown in Figure 14.13(b). Notice that the parameters passed by `main()` to `func1()` are also stored on the stack. If `func1()` were to call any additional functions, storage would be allocated at the new Top of stack as seen in the figure. When `func1()` returns, storage for its local variables is deallocated, and the Top of the stack returns to to position shown in Figure 14.13(c). If `main()` were to call another function, storage would be allocated for that function at the Top shown in the figure. As can be seen, the memory allocated in the stack area is used and reused during program execution. It should be clear that memory allocated in this area will contain garbage values left over from previous usage.

The heap segment provides more stable storage of data for a program; memory allocated in the heap remains in existence for the duration of a program. Therefore, global variables (storage class external), and static variables are allocated on the heap. The memory allocated in the heap area, if initialized to zero at program start, remains zero until the program makes use of it. Thus, the heap area need not contain garbage.

Previous: 14.1.7 Storage Class for Functions
Up: 14.1 Storage Classes
Previous Page: 14.1.7 Storage Class for Functions
Next Page: 14.2 Dynamic Memory Allocation

tep@wiliki.eng.hawaii.edu
Sat Sep 3 07:21:51 HST 1994