

Class 29: Representing Procedure Calls

Held: Wednesday, 13 April 2011

Summary: We consider a basic issue in the implementation of programming languages: How we store information about function calls while the program is running.

Related Pages:

- EBoard.
- Reading: Wikipedia on the Call Stack; Activation Record in *Encyclopedia of Computer Science* (4th ed.).

Notes:

- We'll use about fifteen minutes at the start of class to talk about presentations.
- Readings for Friday: Wikipedia on Evaluation Strategy (http://en.wikipedia.org/wiki/Evaluation_strategy); Encyclopedia of CS on Parameter Passing (<http://portal.acm.org/citation.cfm?id=1074677>).
- EC for Thursday's Convocation [academic]
- EC for Thursday's CS Extra [academic]
- EC for Friday's CS Table - Wolfram TED Talk, http://www.ted.com/talks/stephen_wolfram_computing_a_theory_of_everything.html [academic]
- EC for Bob's Benefit Bash Friday [peer]

Overview:

- Context: Supporting Recursive Functions.
- The Call Stack.
- Making Better Use of the Call Stack.

Recursive Functions

- In the original implementation of function calls, each function had memory allocated at compile time for its parameters and local variables.
- That strategy makes recursion essentially impossible, since recursive calls end up overwriting the local variables.
- Hence, an advance was needed in order to support recursive functions.
- That advance was the call stack. I believe it first appeared in Algol, but I'm probably wrong.
- Idea: Reserve part of memory for information local to each function call. Each time a function is called, allocate more information on the call stack.
- Goes hand-in-hand with that other important memory allocation concept, the heap.

The Call Stack

What needs to be stored when you do a recursive function call?

- I'm not putting the answer here because I want you to come up with the ideas.

Are there ways to make this more efficient?