Stack & Heap

David Croft

Stack and Heap Stack

Stack and Heap memory

David Croft

Coventry University david.croft@coventry.ac.uk

2017



Overview

- 1 Stack and Heap
 - Stack
 - Heap



- Memory model used so far is a simplification.
- Actually two places in memory that variables can go.
 - The stack and the heap.
- Both are just regions of the same physical memory.
 - Are managed differently.



Heap Stack Heap

- When program is run, block of memory is allocated.
 - Called the stack.
- Each program has it's own stack.
 - Each instance.
- As variables created and functions called they are put on the stack.
- When variables are destroyed/functions complete they are removed from the stack.
- Has limited size.
 - Recursive functions can fill the stack if not careful.



```
int main()
     int add( int a, int b)
                                      int var1
         int result = a+b;
                                      int var2
\Rightarrow
                                      int add()
        return result;
                                      int a
                                      int b
                                      int result
     int sub( int a, int b )
         int result = a-b;
\Rightarrow
\Rightarrow
         return result;
     int main()
         int var1 = 42;
\Rightarrow
         int var2 = 1;
\Rightarrow
\Rightarrow
        add(var1,var2);
                                          Stack
\Rightarrow
         sub(var1,var2);
                                                         Heap
\Rightarrow
         return 0;
```



Stack an Heap Stack Heap

- Shared memory between all running programs.
- Very big in comparison to the stack.
- Dangerous, must remember to deallocate our memory.
 - Memory leaks.



```
int main()
     int main()
                                          int variable
\Rightarrow
        int variable = 42;
                                          int *pointer1
        int *pointer1;
                                          int *pointer2
\Rightarrow
        pointer1 = new int[6];
\Rightarrow
        int *pointer2;
\Rightarrow
        pointer2 = new int[3];
\Rightarrow
        delete [] pointer1;
\Rightarrow
        return 0;
\Rightarrow
                                               Stack
```

Heap



Stack

- Fast processors typically have special instructions for dealing with stacks quickly.
- Contiguous everything in one block, easier to know where to put next variable/function.
- Small limited size.
 - Trying too variables will fill stack and cause "stack overflow".

Heap

- Huge relative to the stack.
- Dangerous must remember to deallocate otherwise have memory leaks.



Stack & Heap

David Croft

The End

