	Notes
Functions and the stack	
Michael Nowak	
Texas A&M University	
Acknowledgement: Lecture slides based on those created by Bjarne	
Stroustrup for use with his textbook	
Overview	Notes
Anatomy of a program in memory	
The stack Simplified example	
Where's my program code stored?	
References	
Overview	Notes
Anatomy of a program in memory	
The stack Simplified example	
Where's my program code stored?	
References	

Anatomy of a program in memory

Heap / Free Store The dynamic memory area, where dynamic objects created are store lifetime of heap objects: until explicated or when the program term In classical architectures, the stack and he	
In classical architectures, the stack and he	ed; icitly
towards one another	eap grow
Stores local variables, manages calls; extensively involved in perf computations; lifetime of 'automa objects: persistent until the end o block that declared them	orming itic'

Notes			

Overview

The stack

Simplified example

Notes

The stack

- ▶ During the execution of your programs, the stack manages function calls that are made
- ▶ Each time a function is called, an activation record for that function is pushed (added) to the stack
- ► The activation record is responsible for storing:
 - ► Any necessary house-keeping information (such as return location)
 - The actual arguments passed to the function
 The local variables defined in that function
- ▶ When the function returns to its callee, its activation record is popped (removed) from the stack

Notes			
•			

The stack

- When you compile your code, the compiler examines the variables that will reside on the stack when a respective function is called
- ► The amount of space that is required for each activation record is known up front
- ► When a respective function is called, that amount of memory will be "allocated" on the stack
- ► When that same function has finished executing, the memory associated with its activation record will be "deallocated"
- ► This is why local variables are known as automatic variables: their memory is managed automatically by the function call mechanism

ivotes			

_				
\cap				
u	ve	r٧	ıων	v

Anatomy of a program in memory

The stack

Simplified example

Where's my program code stored?

References

Notes

Simplified example

► When you run your program, an activation record for main is pushed for the stack

Notes			

Simplified example

► When our factorial function fact (int fact(int val)) is called, an activation record for it is pushed (added) to the stack

```
int fact(int val)
{
    int res = 1;
    while(val > 1) {
        res *= val;
        val -= 1;
    }
    return res;
}

int res

int res

int res

int arg

int fact() activation
    record
    int arg

int fval
    int val
    i
```

Notes			

Simplified example

► When our factorial function (int fact(int val)) has finished executing, its activation record is popped (removed) from the stack

Notes			

Overview

Anatomy of a program in memory

The stack

Where's my program code stored?

References

Notes			

Where's my program code stored?	Notes
► The code defining each function (including main) is stored in the code/static region of your program's address space	
► Calling a respective function retrieves the instructions from this region of memory for execution	
Meanwhile, the stack will maintain the following in its activation record:	
 Any necessary house-keeping information (such as return location) Any arguments passed to the function 	
 Any local variables variables that are declared in the function body 	
Overview	Notes
Anatomy of a program in memory	
The stack	
Simplified example	
Where's my program code stored?	
References	
References	Notes
► Lippman, B., Lajoie, Josee, & Moo, B. E. (2016). <i>C++</i> primer (5th ed.). Addison-Wesley.	
 ▶ Stroustrup, B. (2014). Programming: principles and practice using C++ (2nd ed.). Addison-Wesley. 	