Slides + code available at
www.github.com/pkorth/eecs280_recitation

Before we start, please connect to CAEN using the online guide
http://bit.ly/1wYaGwU

EECS 280

Discussion 01: Jan 14, 2015

MichiganEngineering

Logistics

- Patrick Korth
- pkorth@umich.edu (or Piazza)

- Office hours
  - 1695 BBB
  - Monday 1:30 - 3:00 pm
  - Wednesday 1:30 - 3:00 pm

Google

Microsoft

- Slides and code are posted weekly at www.github.com/pkorth/eecs280_recitation

MichiganEngineering

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Logistics

- 50% review, 50% work time
- Suggestion: attend for the review

- Bring a laptop to discussion if possible

- Labs are due Friday at 11:55 pm via CTools
- Group work is encouraged (include names in cpp file)

MichiganEngineering

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Agenda

- Logistics
- Brief review of lecture material
  - Function declaration vs definition
  - Call stacks
  - Recursion
- Lab 01: Linux

Function declaration vs definition

p1.cpp

```
#include "io.h"
int main() {
  PrintHeader();
  // ...
}
```

io.h

```
// …
void PrintHeader(void);
// …
```

io.cpp

```
#include "io.h"
// …

void PrintHeader(void) {
  cout << endl;
  cout << “Month…” <<endl;
  cout << “-----…” <<endl;
}

// …
```

Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```

Stack frame

Represents a function call, its arguments, and local variables.

Call stack

The collection of stack frames we're currently inside.

As we call more and more functions inside of each other, we build up more stack frames.

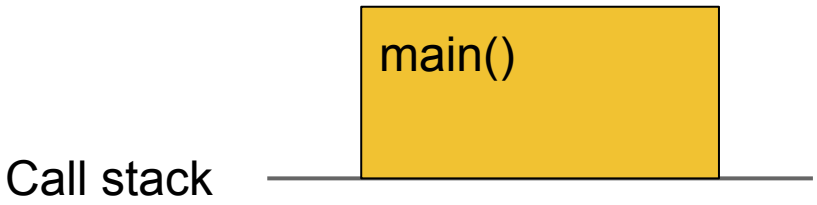
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```

Call stack

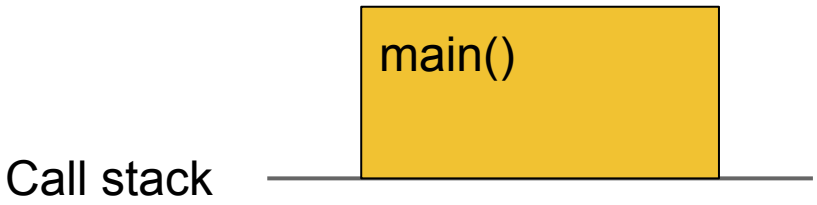
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



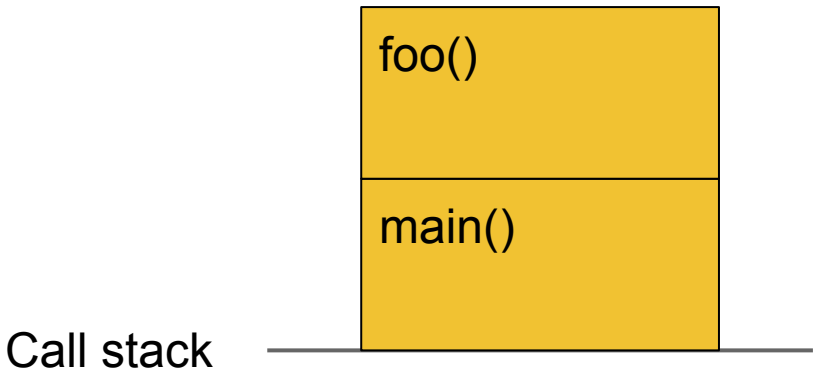
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



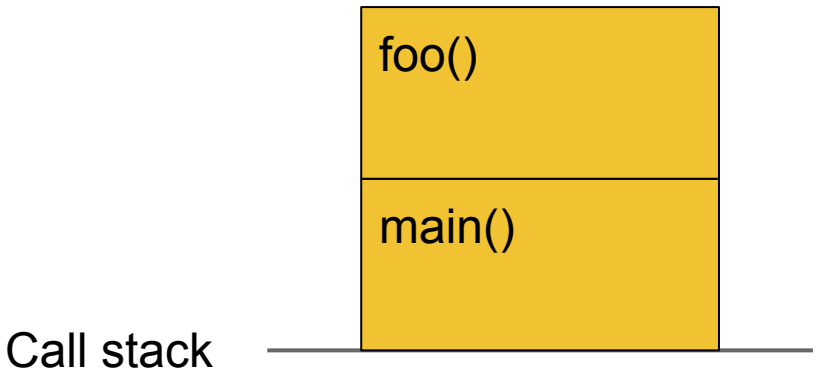
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



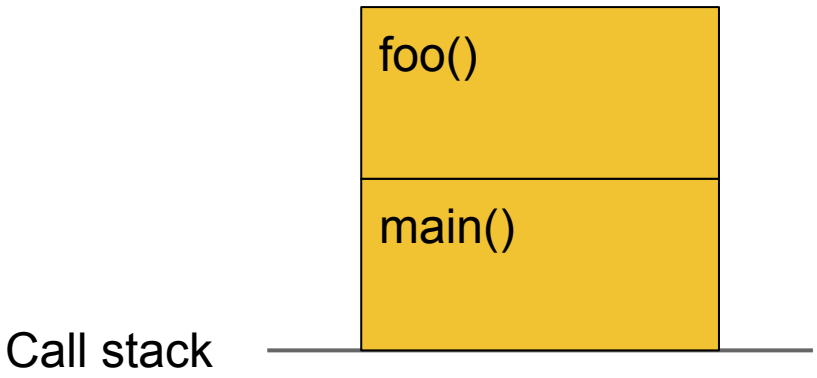
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Call stacks

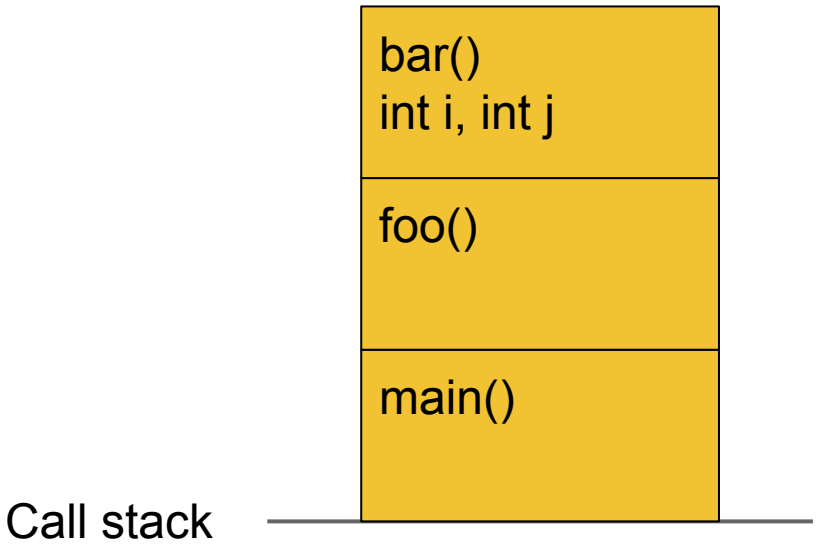
```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Slides + code available at
www.github.com/pkorth/eecs280_recitation

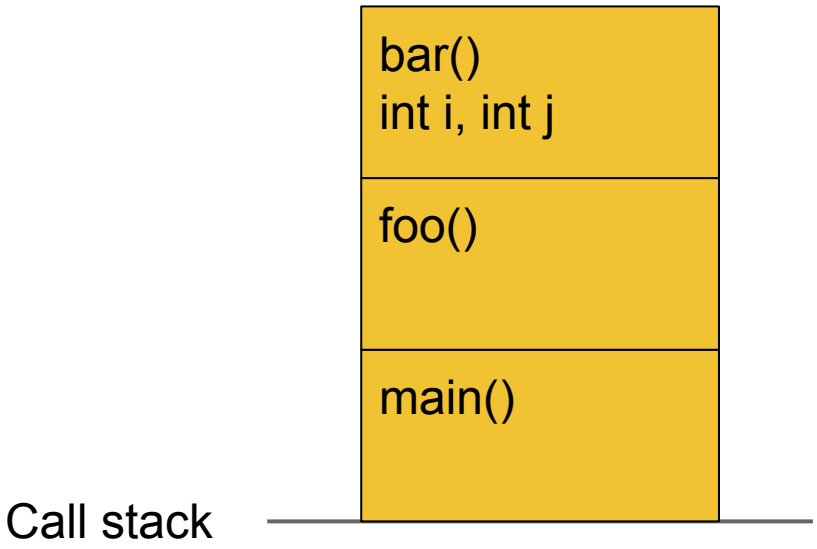
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



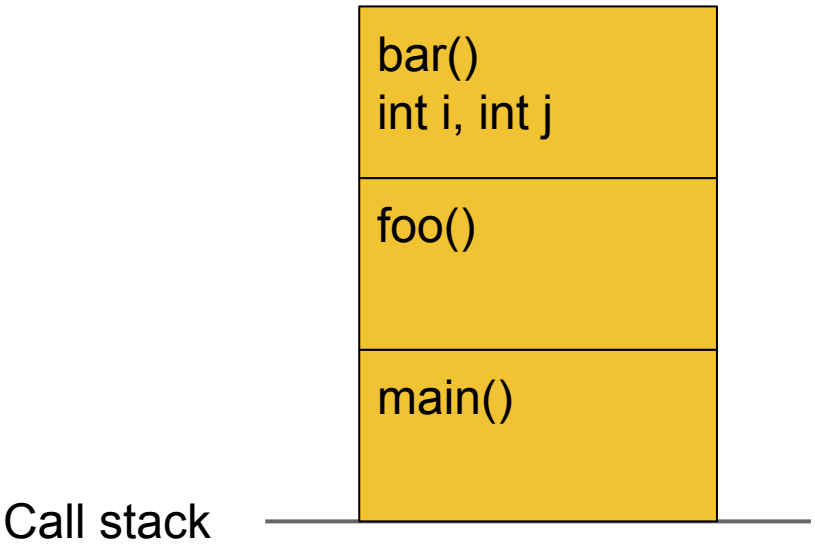
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Call stacks

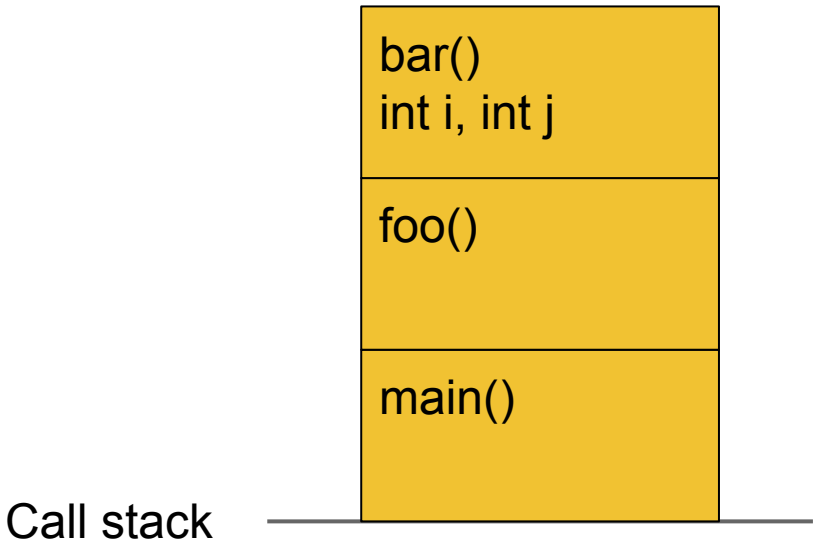
```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Slides + code available at
www.github.com/pkorth/eecs280_recitation

Call stacks

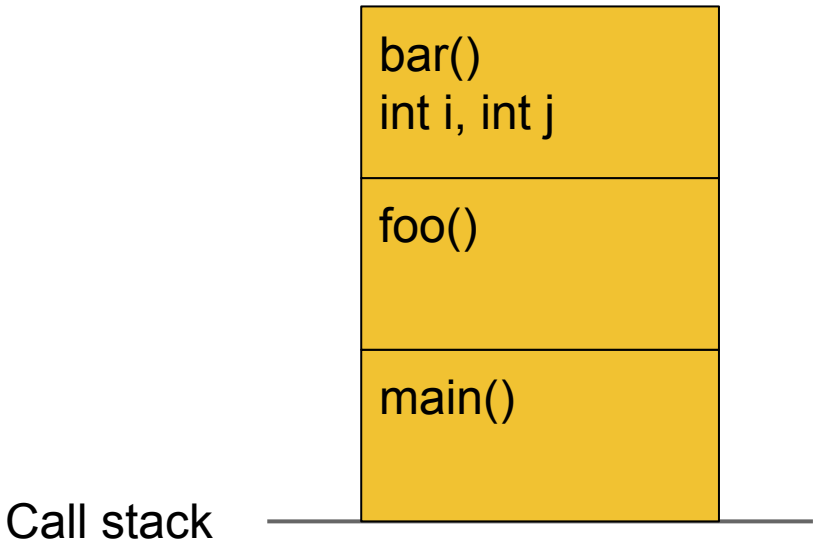
```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Slides + code available at
www.github.com/pkorth/eecs280_recitation

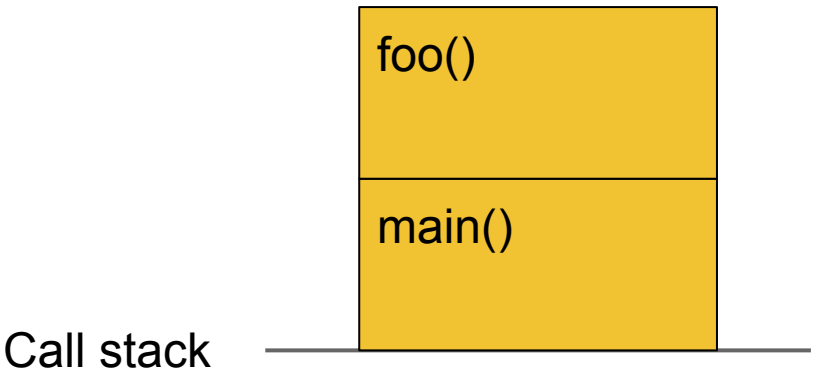
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



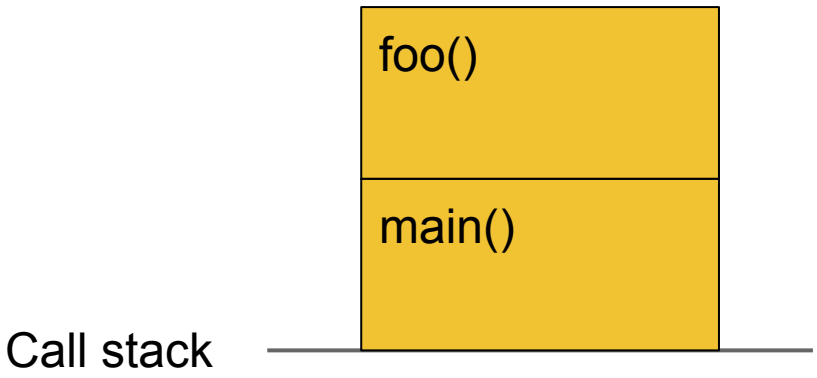
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



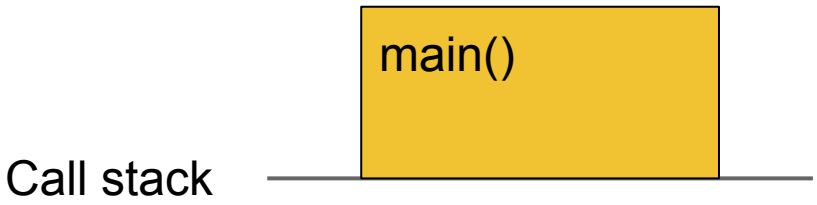
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



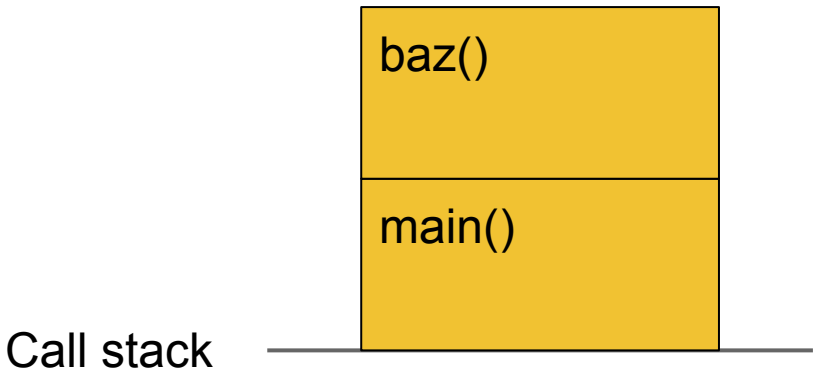
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



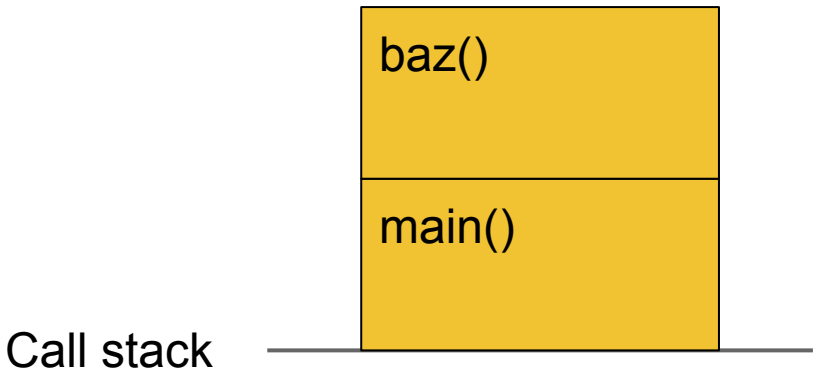
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



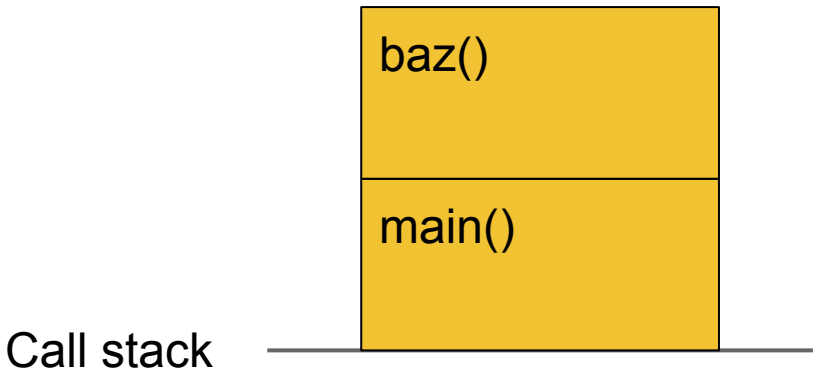
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



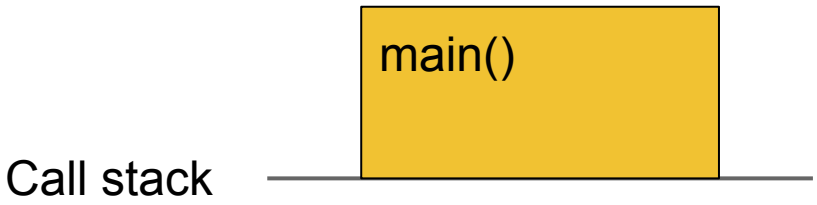
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



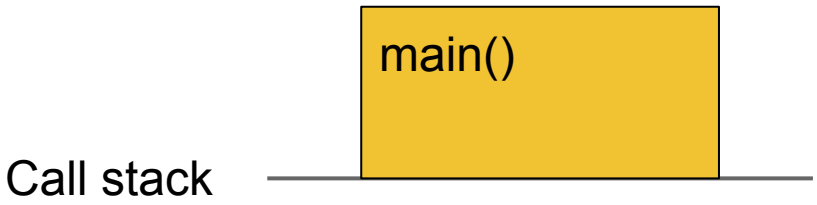
Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```



Call stacks

```
1  int main() {
2    foo();
3    baz();
4    return 0;
5  }
6
7  void foo() {
8    cout << "Foo" << endl;
9    bar(7);
10   cout << "More foo" << endl;
11 }
12
13 void bar(int j) {
14   int i = 5;
15   cout << "Bar" << endl;
16   cout << (i + j) << endl;
17 }
18
19 void baz() {
20   cout << "Baz" << endl;
21 }
```

Call stack

Recursion

A function is said to be **recursive** if it calls itself.

The problem can be defined **in terms of itself**.

This can help simplify some problems by dividing a seemingly large, complicated problem into **smaller subparts** that add together to make the whole.

Recursion

There are two parts to any recursive function:

Base case

The problem can't go any smaller. We know the answer immediately.

Recursive case

Perform a computation for this step, then combine this result with the result of the next smaller step.

Slides + code available at
www.github.com/pkorth/eecs280_recitation

Recursion

```
1  void get_to_destination(here, goal) {
2    if(here == goal)
3      return;
4
5    else
6      dir, dist = ask_for_directions();
7      turn(dir);
8      move_forward(dist);
9
10     get_to_destination(here, goal);
11
12 }
13
```

Route planning

(not real C++)

MichiganEngineering

Recursion

```
1  unsigned int factorial(unsigned int n) {
2    if (n == 0) {
3      return 1;
4
5    } else {
6      return n * factorial(n - 1);
7    }
8  }
9
10
11
12
13
```

Factorial

0! = 1

1! = 1

n! = n x (n - 1)!

4! = 4 x 3 x 2 x 1 = 24

Lab 01

Goal

Learn the basics of programming on Linux

Tasks

1. Connect to CAEN Linux and create a lab01 directory
2. Download the lab files
3. Edit the code a bit
4. Compile and run the code
5. Use gdb to find and fix the bug