Week 5 Review

Rob Bowden

Stack

- One stack frame per “active” function call
- Stores local variables and passed-in arguments

Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```

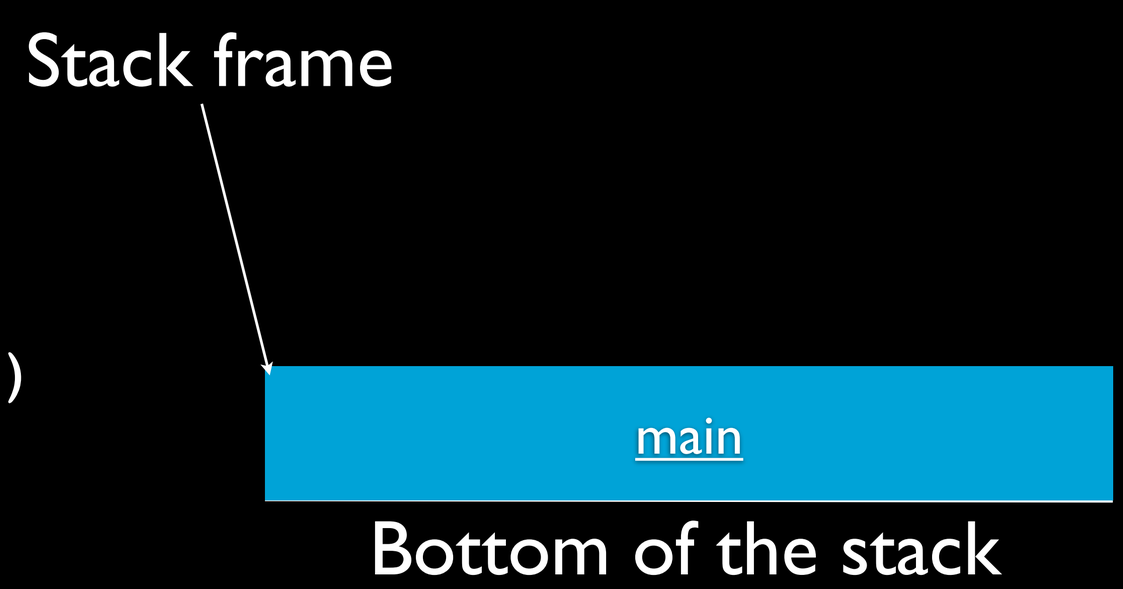
Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```

Bottom of the stack

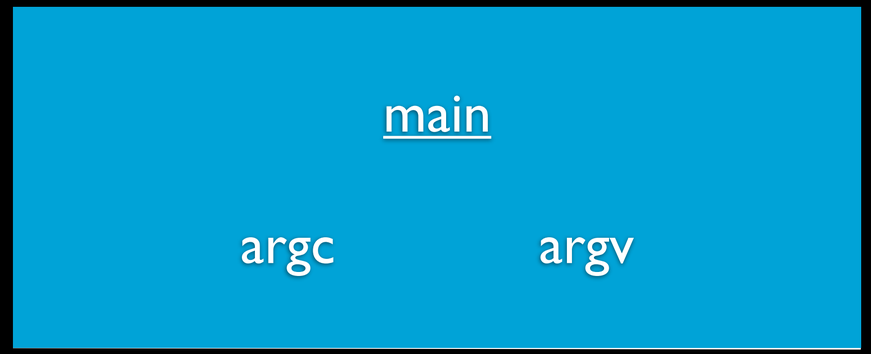
Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Stack

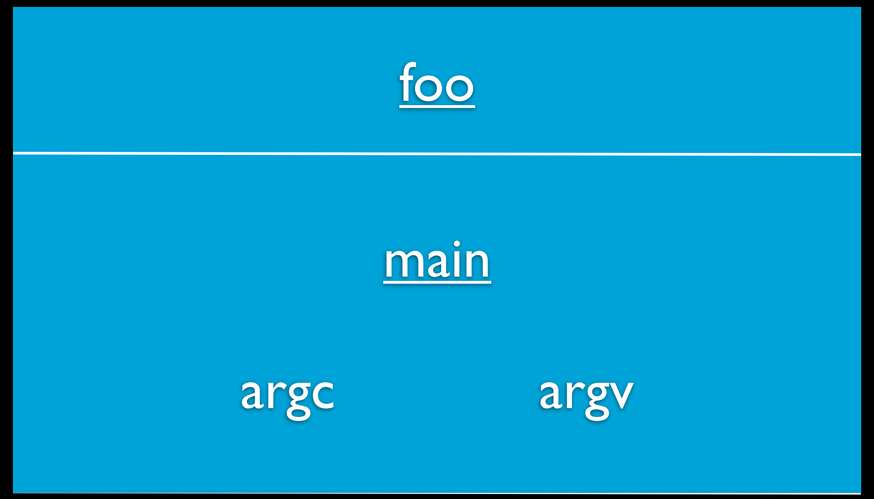
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

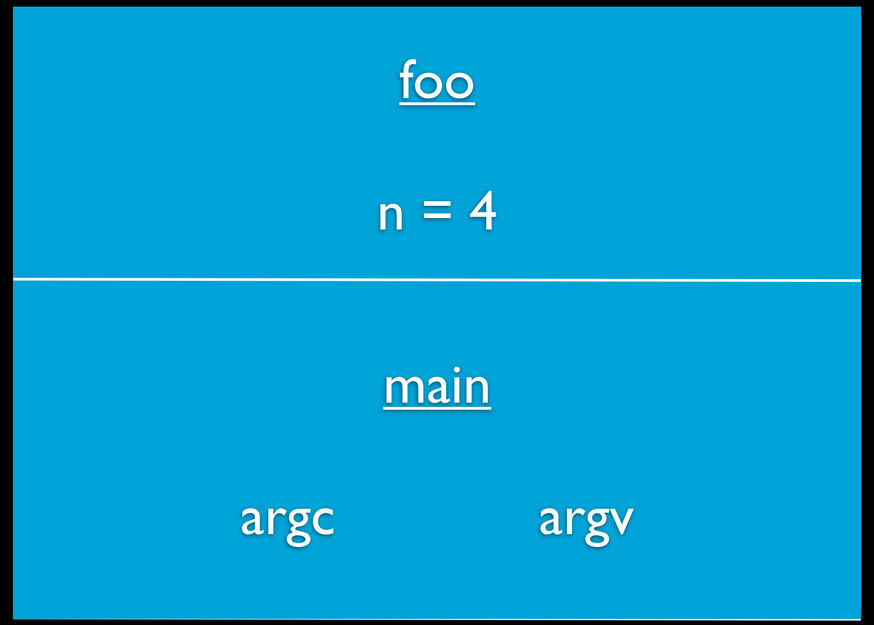
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

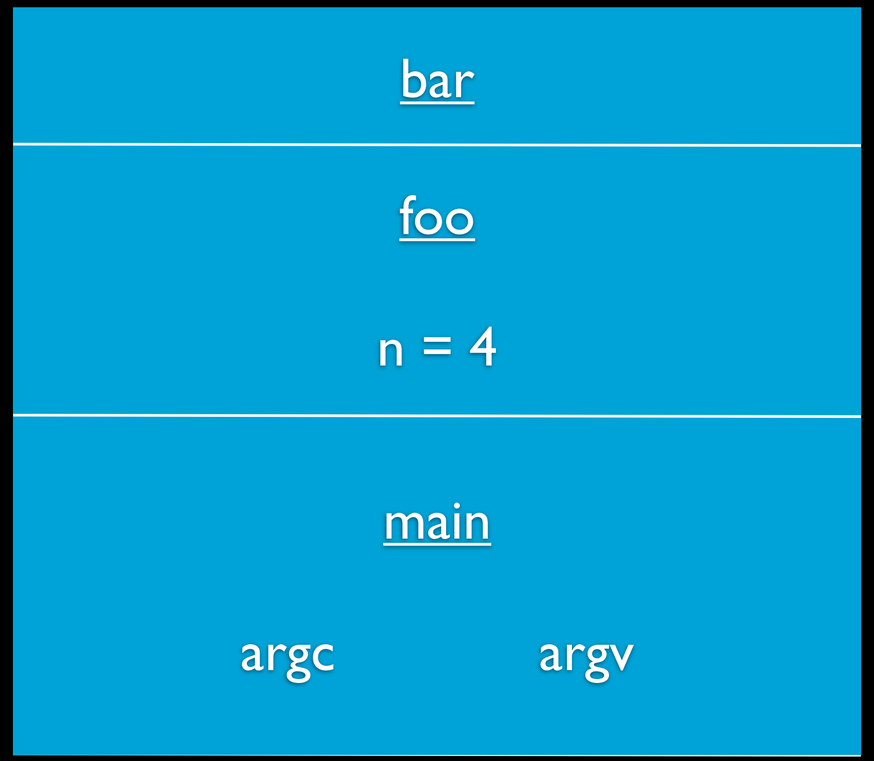
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

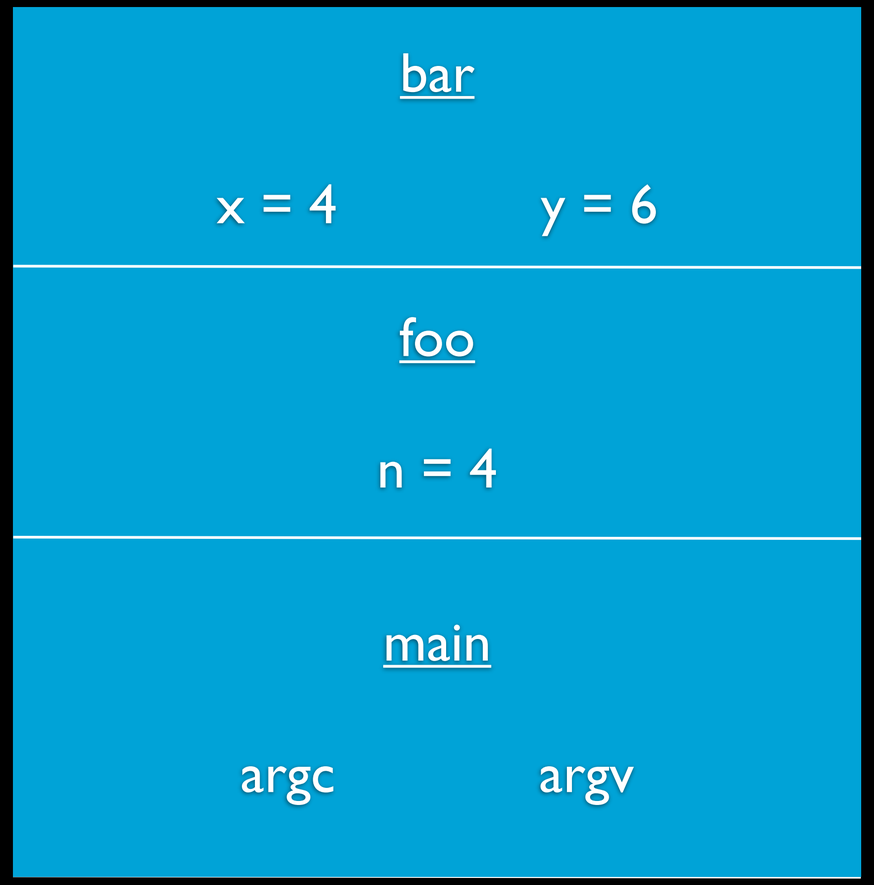
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

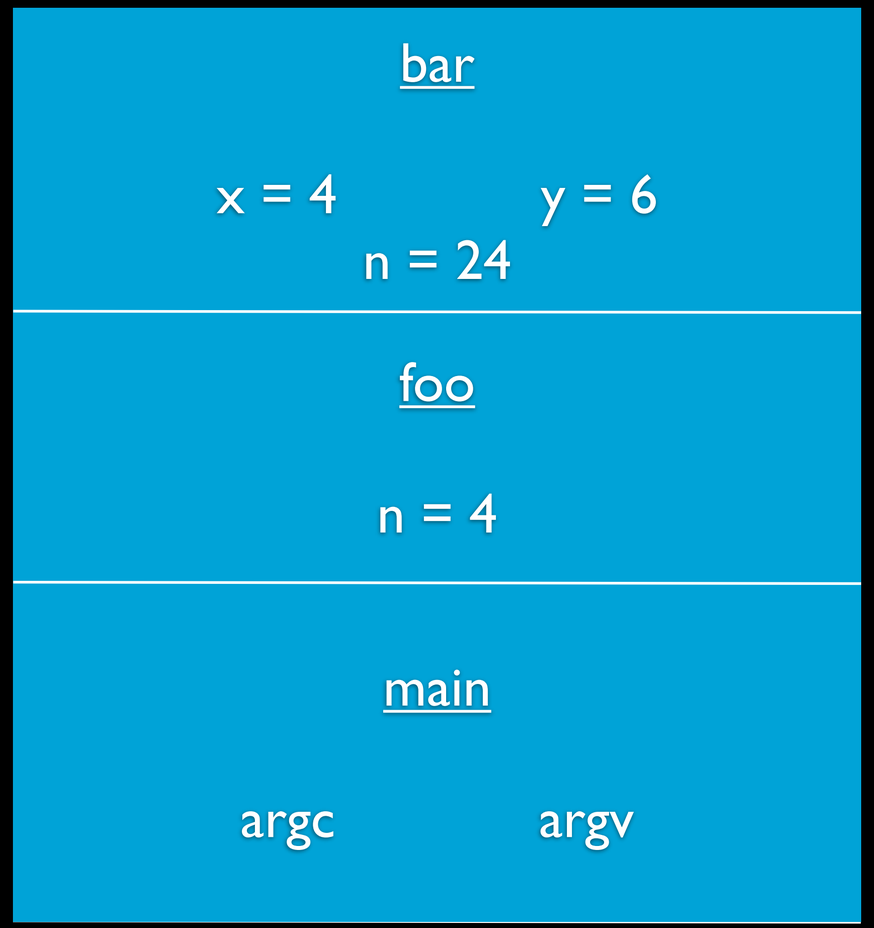
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

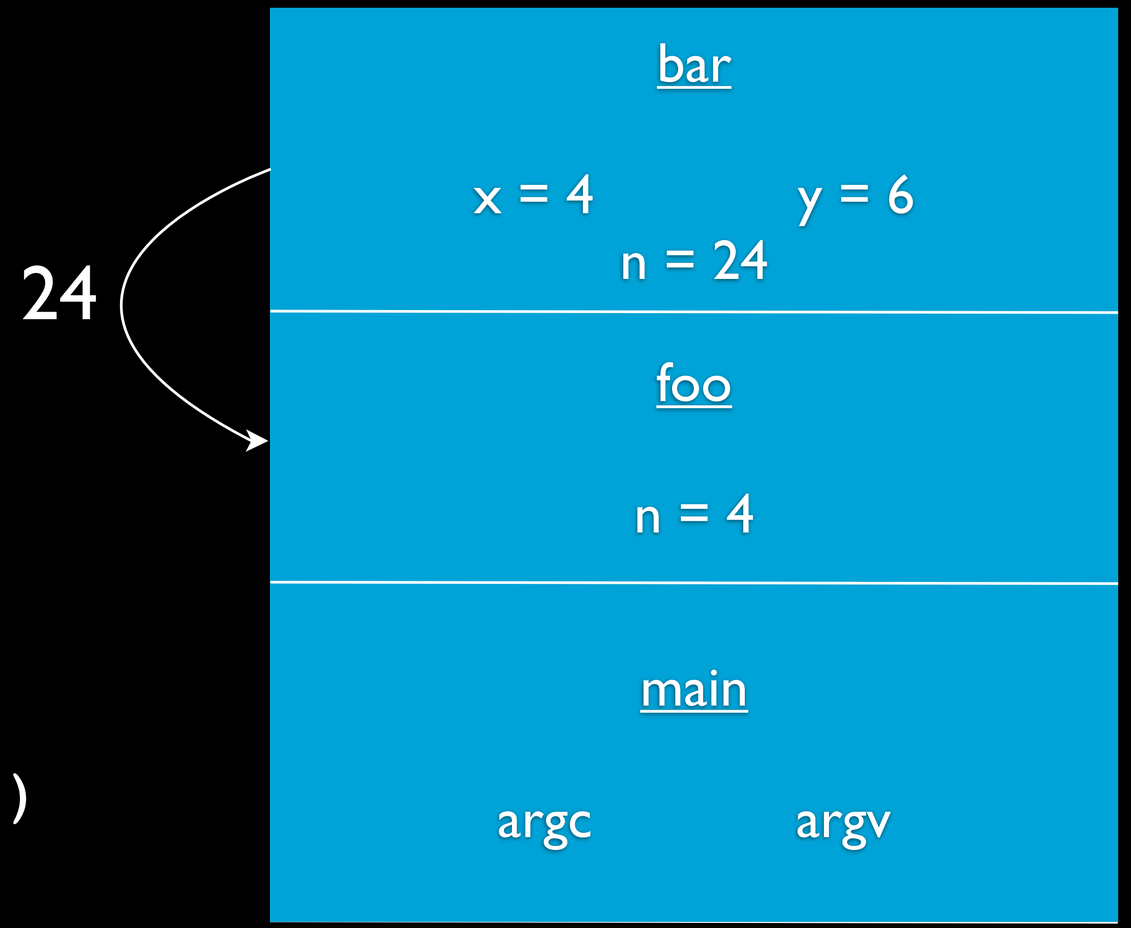
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

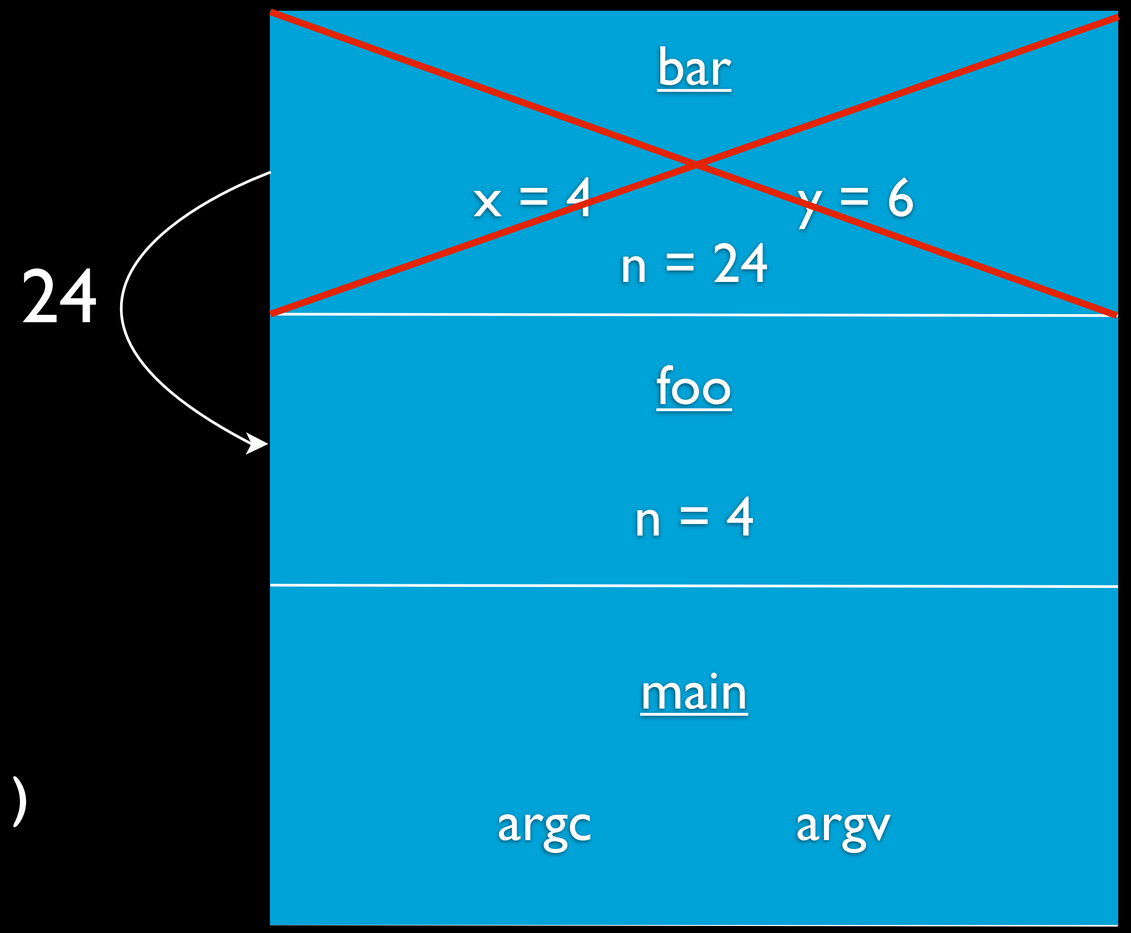
Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Stack

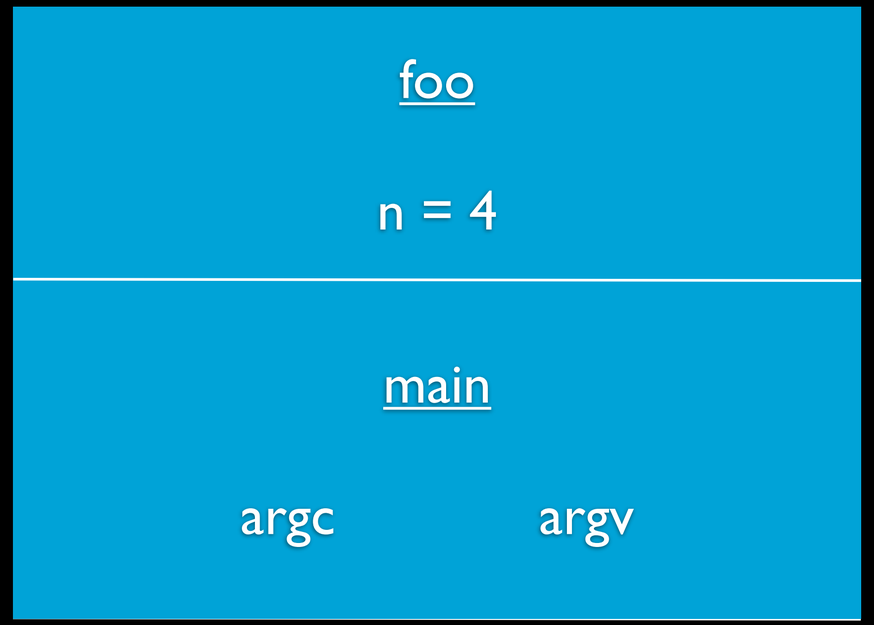
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

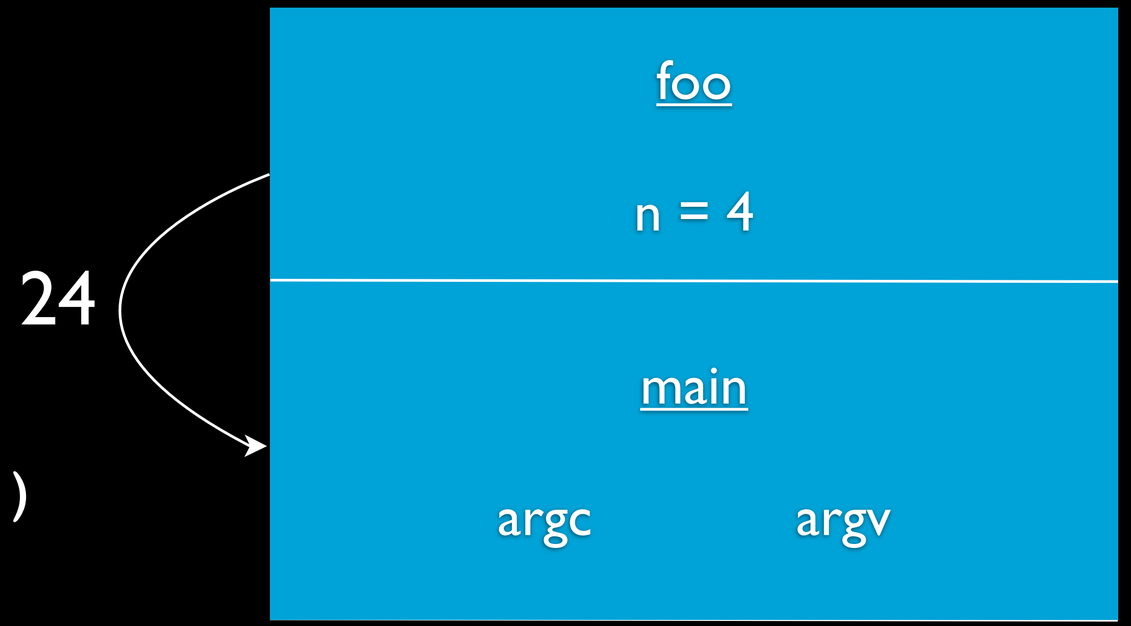
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

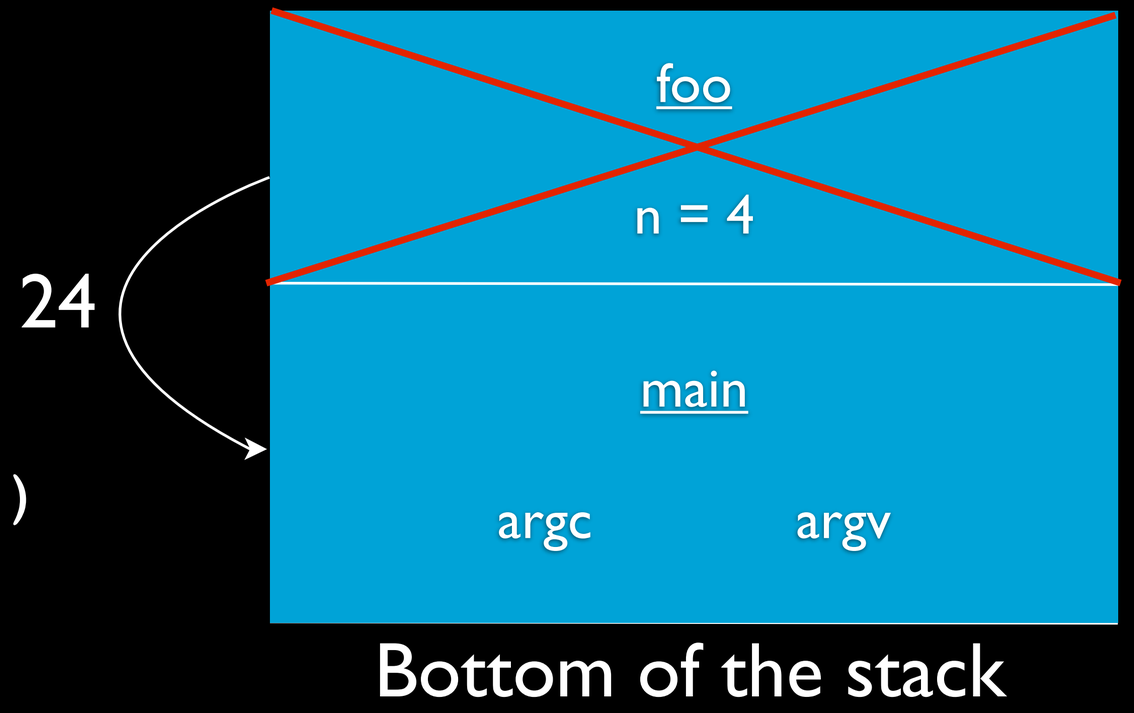
Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



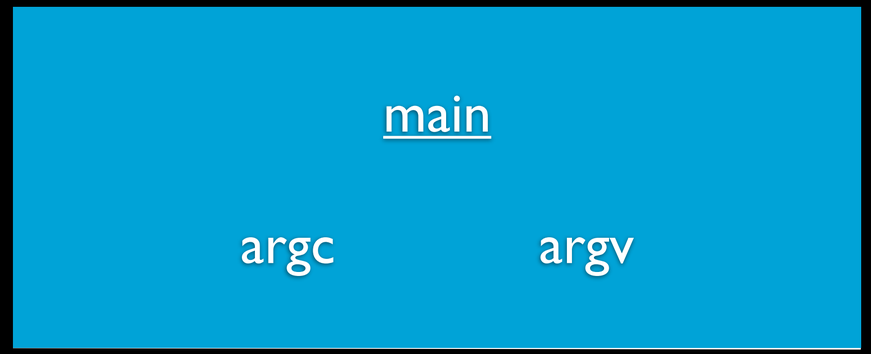
Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Stack

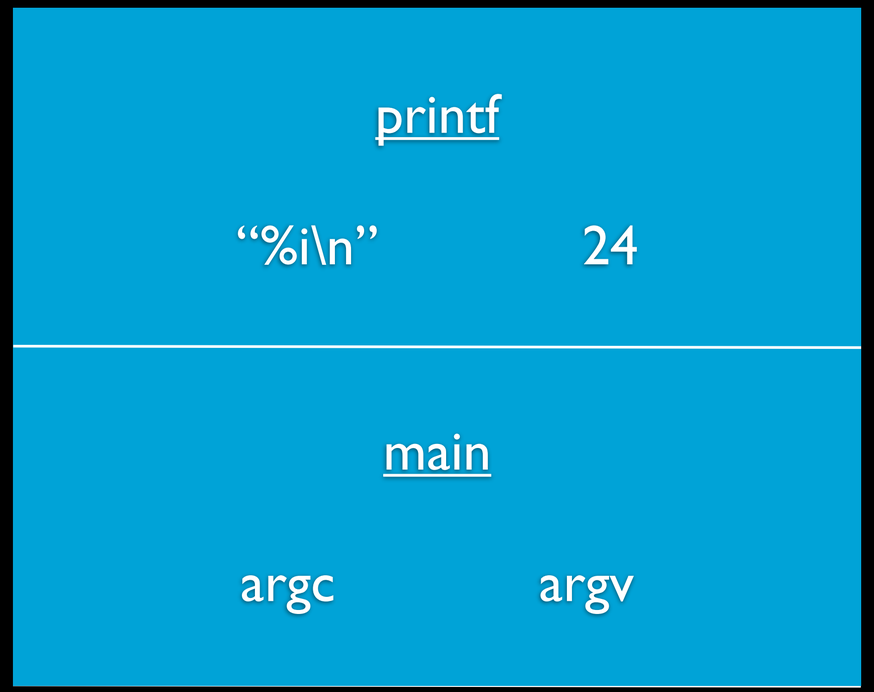
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

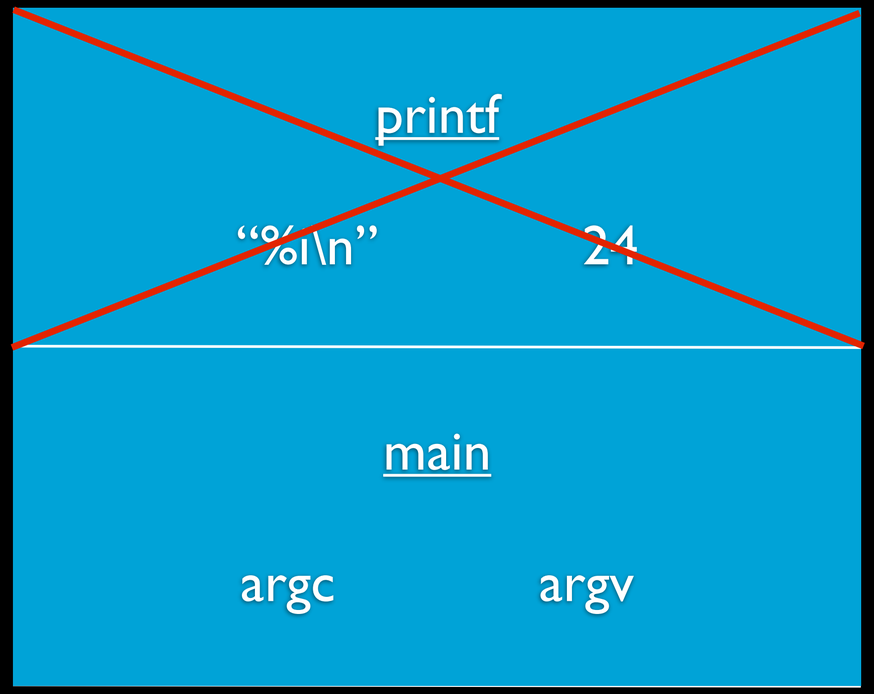
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

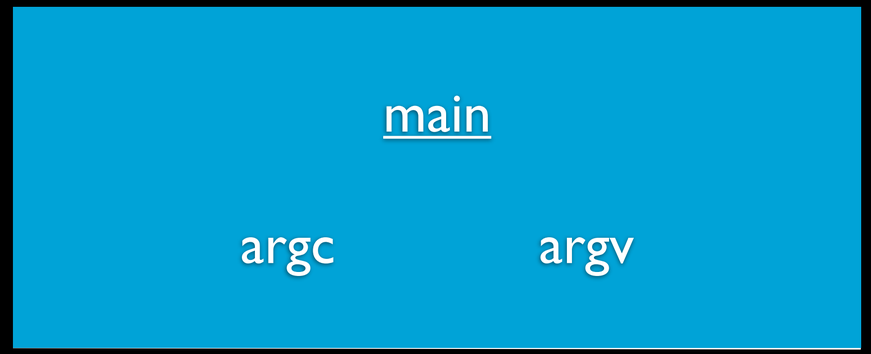
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

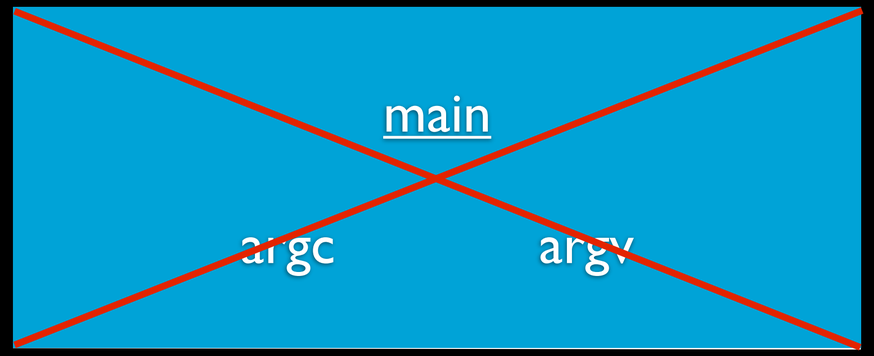
```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```



Bottom of the stack

Stack

```
int bar(int x, int y)
{
    int n = x * y;
    return n;
}

int foo(int n)
{
    return bar(n, n + 2);
}

int main(int argc, string argv[])
{
    printf("%i\n", foo(4));
}
```

Bottom of the stack

Heap

- Dynamic memory management
- malloc and free

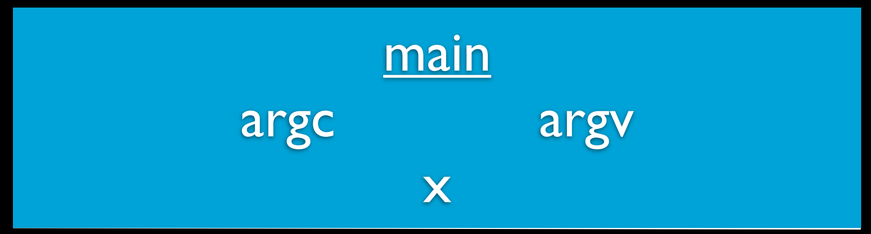
Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Top of the heap

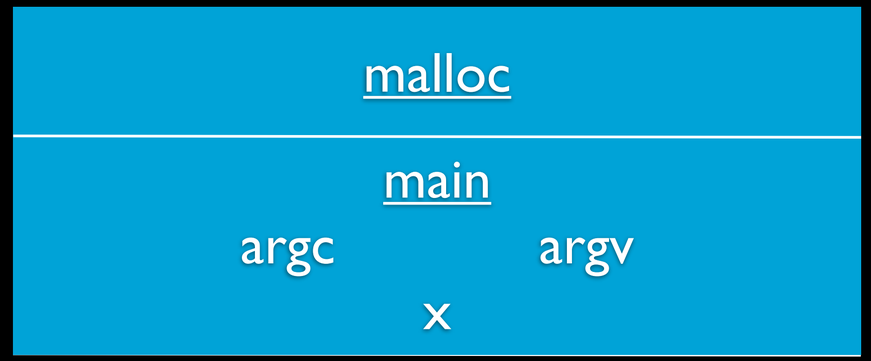


Bottom of the stack

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Top of the heap



Bottom of the stack

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Top of the heap

Bottom of the stack

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

0x123

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

0x123

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

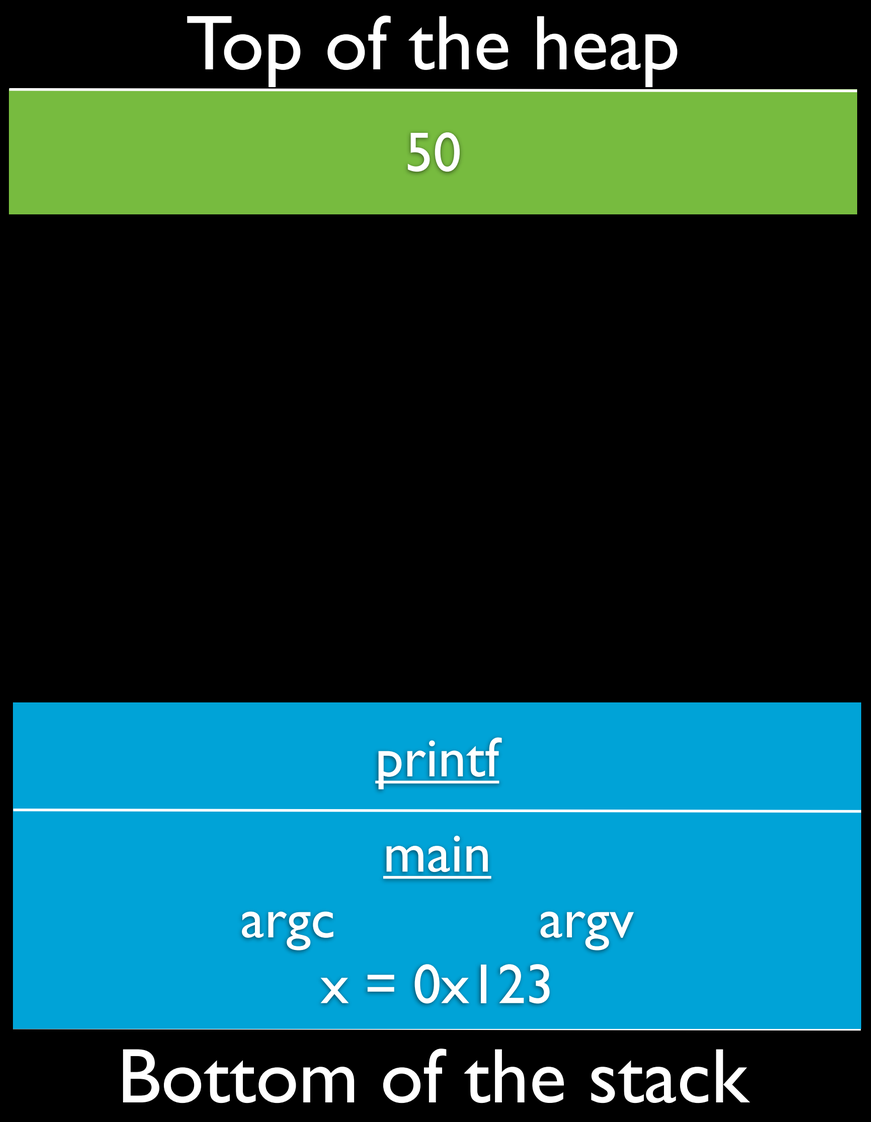
```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```



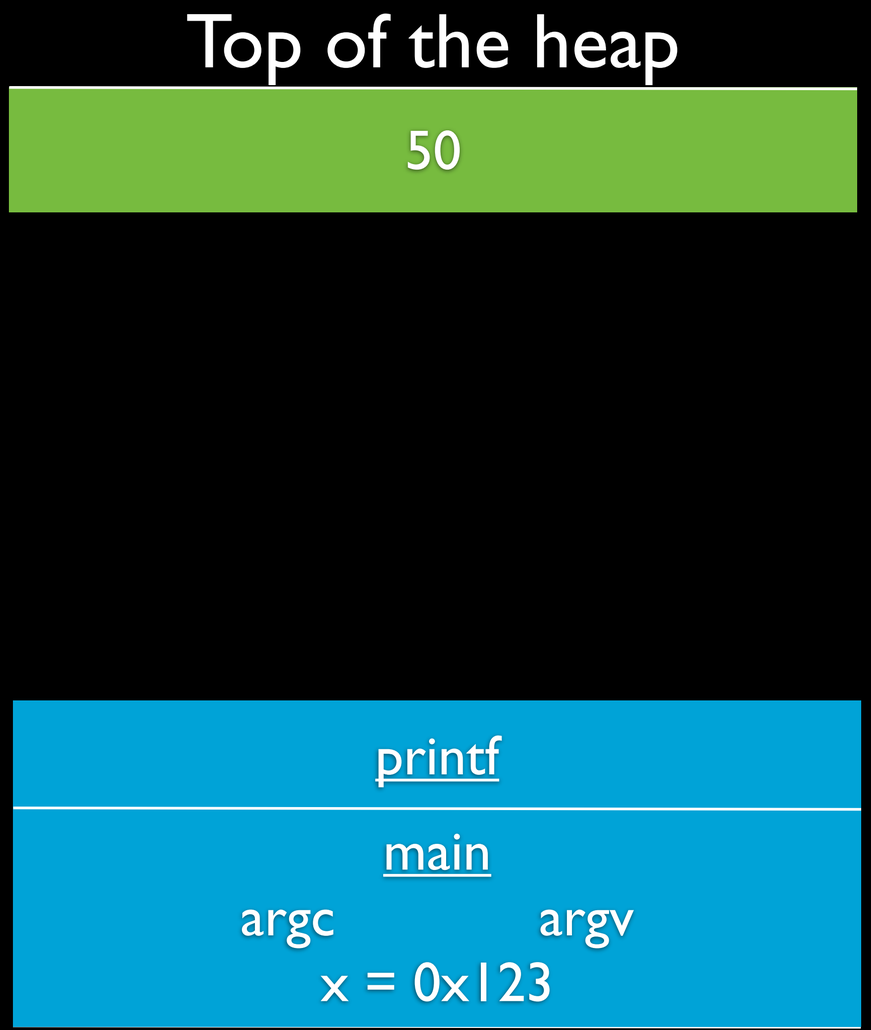
Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

0x123



Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

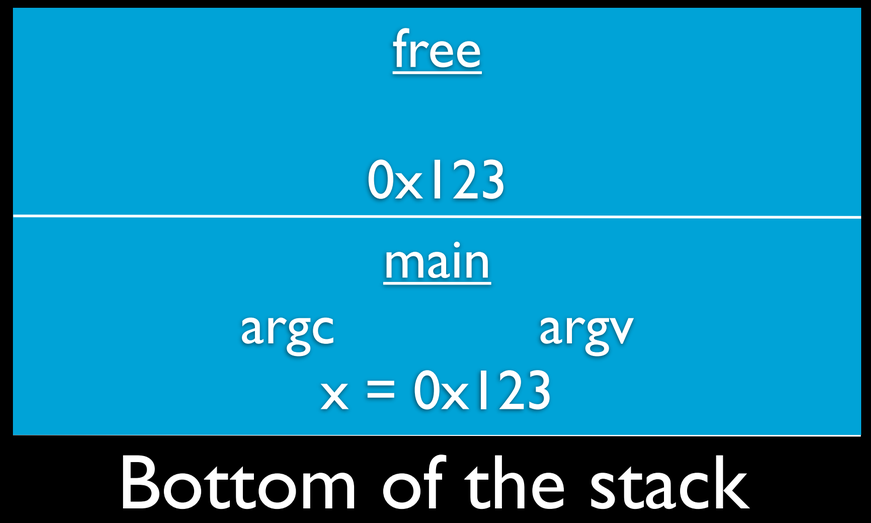
```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Heap

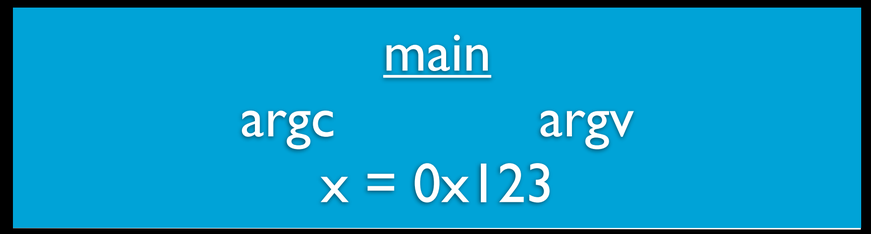
```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```



Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Top of the heap



Bottom of the stack

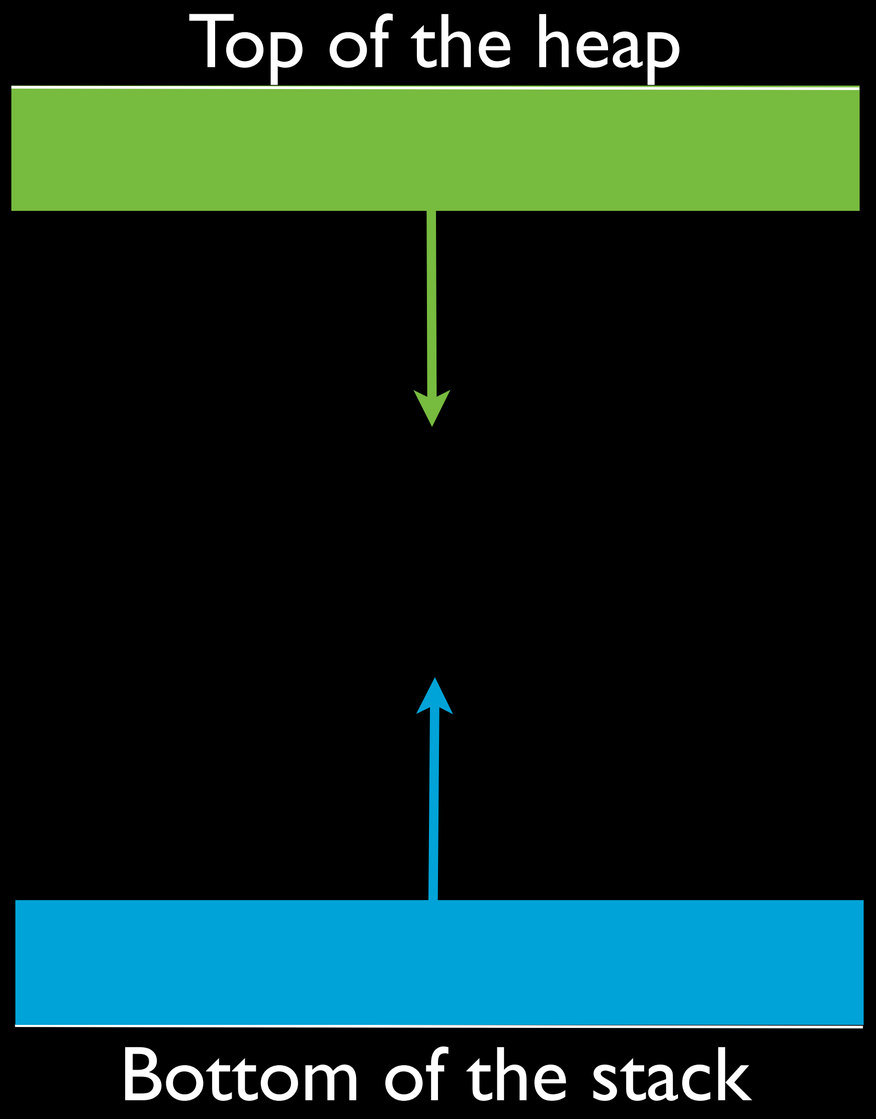
Heap

```
int main(int argc, string argv[])
{
    int* x = malloc(sizeof(int));
    if (x == NULL)
    {
        return 1;
    }
    *x = 50;
    printf("%i\n", *x);
    printf("%p\n", x);
    free(x);
    return 0;
}
```

Top of the heap

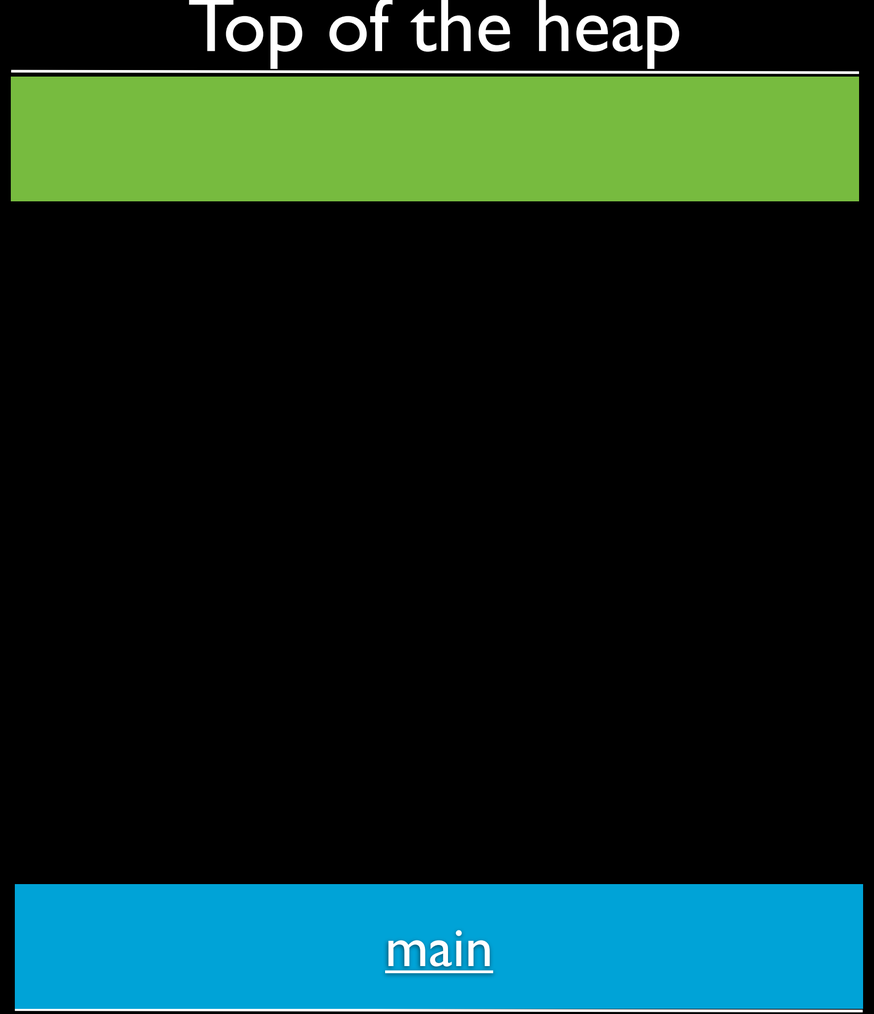
Bottom of the stack

Stack overflow



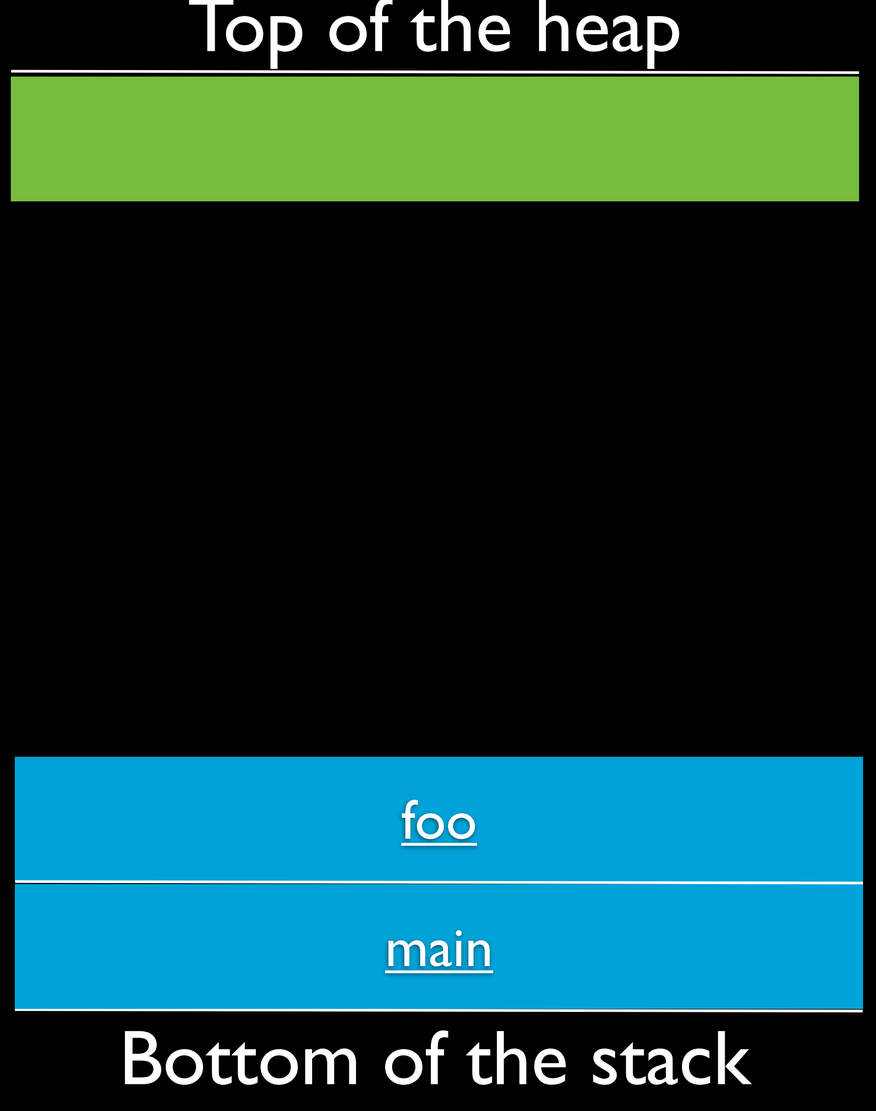
Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```



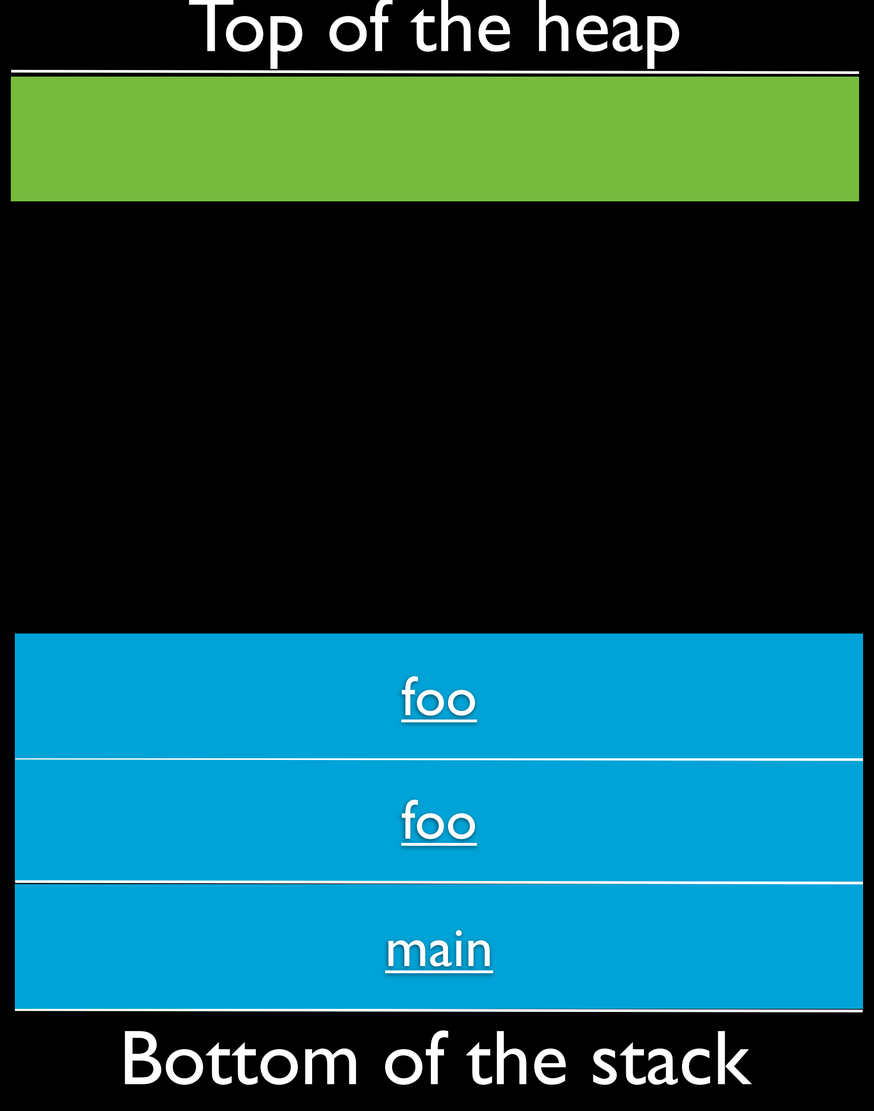
Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```



Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```



Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```

Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```

Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```

Stack overflow

```
void foo(void)
{
    foo();
}

int main(int argc, string argv[])
{
    foo();
}
```

Compilation

- Pre-processing (#)
- Compiling (C => Assembly)
- Assembling (Assembly => Binary)
- Linking (Binary => Executable <= Binary)