

# EECS 10: Computational Methods in Electrical and Computer Engineering

## Lecture 14

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering  
Electrical Engineering and Computer Science  
University of California, Irvine

## Lecture 14: Overview

- Course Administration
  - Reminder: Midterm course evaluation
- Functions
  - Hierarchy of functions
    - Example `cylinder.c`
  - Function call graph
  - Function call trace
  - Function call stack

## Course Administration

- Midterm Course Evaluation
  - Open until tonight!
  - Monday, Oct. 25, 9am – Friday, Oct. 29, 9pm
  - Online via EEE Evaluation application
- Feedback from students to instructors
  - Completely voluntary
  - Completely anonymous
  - Very valuable
    - Help to improve this class!
- Mandatory Final Course Evaluation
  - expected for week 10 (TBA)

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

3

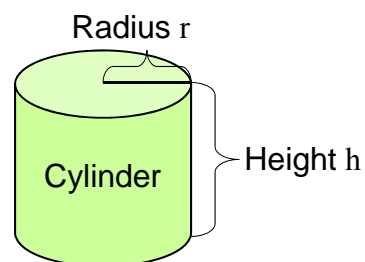
## Functions

- Hierarchy of Functions
  - functions call other functions

- Example:

### Cylinder calculations

- given radius and height
- calculate surface and volume



- Circle constant  $\pi = 3.14159265\dots$
- Circle perimeter  $f_p(r) = 2 \times \pi \times r$
- Circle area  $f_a(r) = \pi \times r^2$
- Cylinder surface  $f_s(r, h) = f_p(r) \times h + 2 \times f_a(r)$
- Cylinder volume  $f_v(r, h) = f_a(r) \times h$

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

4

## Functions

- Program example: `Cylinder.c` (part 1/3)

```

/* Cylinder.c: cylinder functions */
/* author: Rainer Doemer */
/* modifications: */
/* 10/25/05 RD initial version */

#include <stdio.h>

/* cylinder functions */

double pi(void)
{
    return(3.1415927);
}

double CircleArea(double r)
{
    return(pi() * r * r);
}
...

```

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

5

## Functions

- Program example: `Cylinder.c` (part 2/3)

```

...
double CirclePerimeter(double r)
{
    return(2 * pi() * r);
}

double Surface(double r, double h)
{
    double side, lid;
    side = CirclePerimeter(r) * h;
    lid = CircleArea(r);
    return(side + 2*lid);
}

double Volume(double r, double h)
{
    return(CircleArea(r) * h);
}
...

```

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

6

## Functions

- Program example: `Cylinder.c` (part 3/3)

```

...
/* main function */
int main(void)
{
    double r, h, s, v;

    /* input section */
    printf("Please enter the radius: ");
    scanf("%lf", &r);
    printf("Please enter the height: ");
    scanf("%lf", &h);

    /* computation section */
    s = Surface(r, h);
    v = Volume(r, h);

    /* output section */
    printf("The surface area is %f.\n", s);
    printf("The volume is %f.\n", v);

    return 0;
} /* end of main */

```

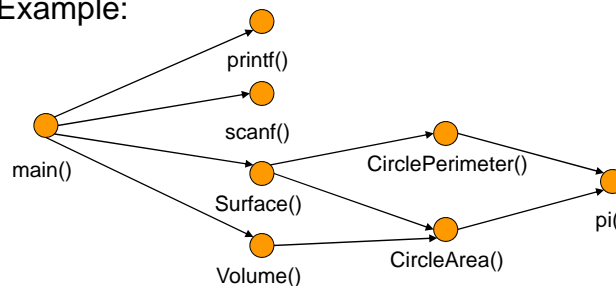
EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

7

## Function Call Graph

- Graphical representation of function calls
  - Directed Graph
    - Nodes: Functions
    - Edges: Function calls
  - Shows dependencies among functions
  - Example:



EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

8

## Function Call Trace

- Sequence of function calls
  - Shows execution order of functions at run-time
- Example:
  - main()
    - printf()
    - scanf()
    - printf()
    - scanf()
    - Surface()
      - CirclePerimeter()
        - » pi()
      - CircleArea()
        - » pi()
    - Volume()
      - CircleArea()
        - » pi()
    - printf()
    - printf()

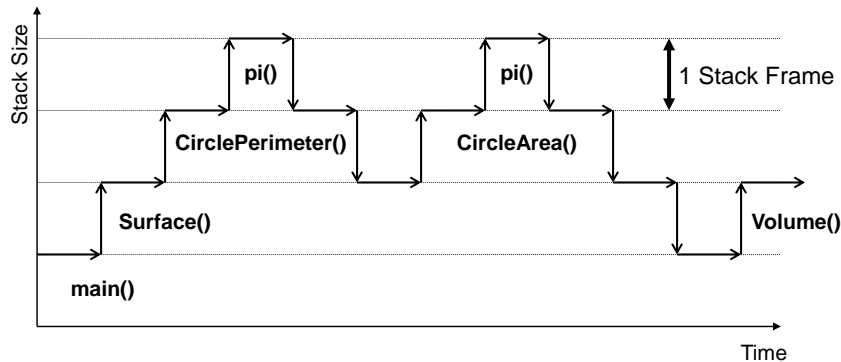
EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

9

## Function Call Stack

- Stack Frames
  - Keep track of active function calls
    - Stack grows by one frame with each function call
    - Stack shrinks by one frame with each completed function



EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

10

## Function Call Stack

- Stack Frames
  - Keep track of active function calls
    - Stack grows by one frame with each function call
    - Stack shrinks by one frame with each completed function

The diagram shows a vertical axis for 'Stack Size' and a horizontal axis for 'Time'. The stack grows from bottom to top as functions are called. The sequence of frames is: main(), Surface(), CirclePerimeter(), pi(), CircleArea(), pi(), and Volume(). Blue arrows point upwards for each call and downwards for each return. A vertical double-headed arrow on the right indicates the height of one stack frame.

EECS10: Computational Methods in ECE, Lecture 14 (c) 2010 R. Doemer 11

## Function Call Stack

- Stack Frames
  - Keep track of active function calls
    - Stack grows by one frame with each function call
    - Stack shrinks by one frame with each completed function

This diagram is identical to the one above, but uses red arrows to highlight the return phase of each function call. Blue arrows point up for calls, and red arrows point down for returns. The sequence of frames remains: main(), Surface(), CirclePerimeter(), pi(), CircleArea(), pi(), and Volume().

EECS10: Computational Methods in ECE, Lecture 14 (c) 2010 R. Doemer 12

## Debugging

- Source-level Debugger `gdb`
  - Basic `gdb` commands
    - `run`
      - starts the execution of the program in the debugger
    - `break function_name (or line_number)`
      - inserts a breakpoint; program execution will stop at the breakpoint
    - `cont`
      - continues the execution of the program in the debugger
    - `list from_line_number,to_line_number`
      - lists the current or specified range of `line_numbers`
    - `print variable_name`
      - prints the current value of the variable `variable_name`
    - `next`
      - executes the next statement (one statement at a time)
    - `quit`
      - exits the debugger (and terminates the program)
    - `help`
      - provides helpful details on debugger commands

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

13

## Debugging

- Source-level Debugger `gdb` (continued)
  - Additional `gdb` commands
    - `step`
      - steps into a function call
    - `finish`
      - continues execution until the current function is finished
    - `where`
      - shows where in the function call hierarchy you are
      - prints a *back trace* of current *stack frames*
    - `up`
      - steps up one stack frame (up into the caller)
    - `down`
      - steps down one stack frame (down into the callee)
    - `info locals`
      - lists the local variables in the current function (current stack frame)
    - `info scope function_name`
      - lists the variables in scope of the `function_name`

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

14

## Function Call Stack

- Navigating Stack Frames in the Debugger
  - *step*: execute and step into a function call
  - *up*, *down*: navigate stack frames
  - *finish*: resume execution until the end of the current function

Stack Size

Time

EECS10: Computational Methods in ECE, Lecture 14 (c) 2010 R. Doemer 15

## Functions

- Example session: `Cylinder.c`

```

% vi Cylinder.c
% gcc Cylinder.c -o Cylinder -Wall -ansi -g
% gdb Cylinder
GNU gdb 6.3
(gdb) break 55
Breakpoint 1 at 0x108d0: file Cylinder.c, line 55.
(gdb) run
Starting program: /users/faculty/doemer/eecs10/Cylinder/Cylinder
Please enter the radius: 10
Please enter the height: 10
Breakpoint 1, main () at Cylinder.c:56
56     s = Surface(r, h);
(gdb) step
Surface (r=10, h=10) at Cylinder.c:31
31     side = CirclePerimeter(r) * h;
(gdb) step
CirclePerimeter (r=10) at Cylinder.c:24
24     return(2 * pi() * r);
...
    
```

EECS10: Computational Methods in ECE, Lecture 14 (c) 2010 R. Doemer 15



## Functions

- Example session: `Cylinder.c`

```
(gdb) step
pi () at Cylinder.c:14
14      return(3.1415927);
(gdb) where
#0  pi () at Cylinder.c:14
#1  0x000107bc in CirclePerimeter (r=10) at Cylinder.c:24
#2  0x000107f8 in Surface (r=10, h=10) at Cylinder.c:31
#3  0x000108e0 in main () at Cylinder.c:56
(gdb) up
#1  0x000107bc in CirclePerimeter (r=10) at Cylinder.c:24
24      return(2 * pi() * r);
(gdb) up
#2  0x000107f8 in Surface (r=10, h=10) at Cylinder.c:31
31      side = CirclePerimeter(r) * h;
(gdb) up
#3  0x000108e0 in main () at Cylinder.c:56
56      s = Surface(r, h);
...
```

EECS10: Computational Methods in ECE, Lecture 14

(c) 2010 R. Doemer

17

## Functions

- Example session: `Cylinder.c`

```
(gdb) down
#2  0x000107f8 in Surface (r=10, h=10) at Cylinder.c:31
31      side = CirclePerimeter(r) * h;
(gdb) down
#1  0x000107bc in CirclePerimeter (r=10) at Cylinder.c:24
24      return(2 * pi() * r);
(gdb) down
#0  pi () at Cylinder.c:14
14      return(3.1415927);
(gdb) finish
Run till exit from #0  pi () at Cylinder.c:14
0x000107bc in CirclePerimeter (r=10) at Cylinder.c:24
24      return(2 * pi() * r);
Value returned is $1 = 3.1415926999999999
(gdb) finish
Run till exit from #0  CirclePerimeter (r=10) at Cylinder.c:24
0x000107f8 in Surface (r=10, h=10) at Cylinder.c:31
31      side = CirclePerimeter(r) * h;
...
EE
```

## Functions

- Example session: `Cylinder.c`

```
Value returned is $2 = 62.831854
(gdb) next
32      lid = CircleArea(r);
(gdb) step
CircleArea (r=10) at Cylinder.c:19
19      return(pi() * r * r);
(gdb) finish
Run till exit from #0  CircleArea (r=10) at Cylinder.c:19
0x00010818 in Surface (r=10, h=10) at Cylinder.c:32
32      lid = CircleArea(r);
Value returned is $3 = 314.15926999999999
(gdb) cont
Continuing.
The surface area is 1256.637080.
The volume is 3141.592700.
Program exited normally.
(gdb) quit
%
```