Introduction to Computer Science and Programming in C

Session 4: September 11, 2008

Columbia University

Announcements

- Reminder: Homework 1 is out. Due 9/23
- Additional TA: Peter Lu. Office hours TBA.
  UNI: yl2505
- Use the message boards.
  Counts as class participation.

Review

- Unix/cunix demo
  - PuTTY fix
- Hello World
- Submission procedure
- Breaking down Hello World (cut off)

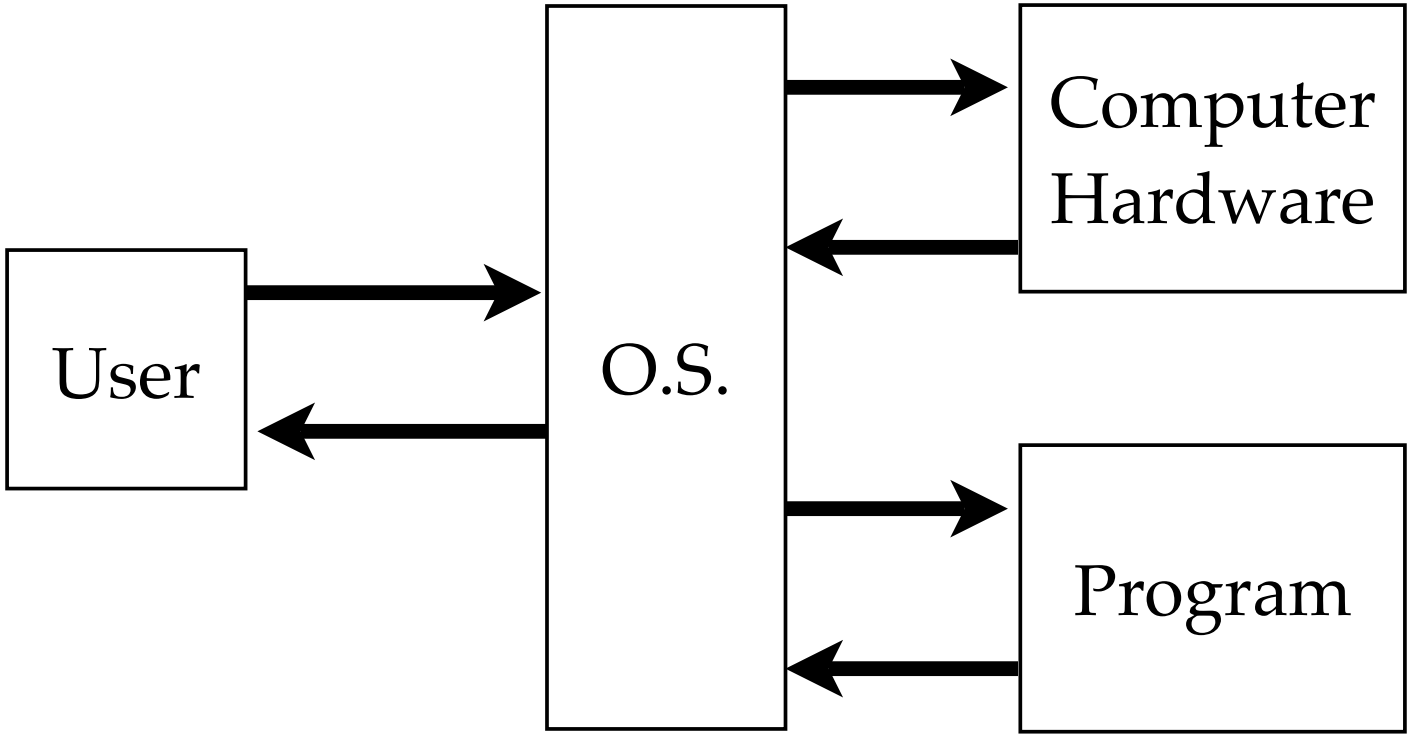
PuTTY Fix

- http://www.cs.columbia.edu/~bert/courses/1003/putty/

Today

- Breaking down Hello World (continued)
- Variables and basic types

Computer Programs Illustrated



```
/*
Bert Huang. My first program.
*/


#include <stdio.h>


int main()
{

   printf("Hello, World!\n");



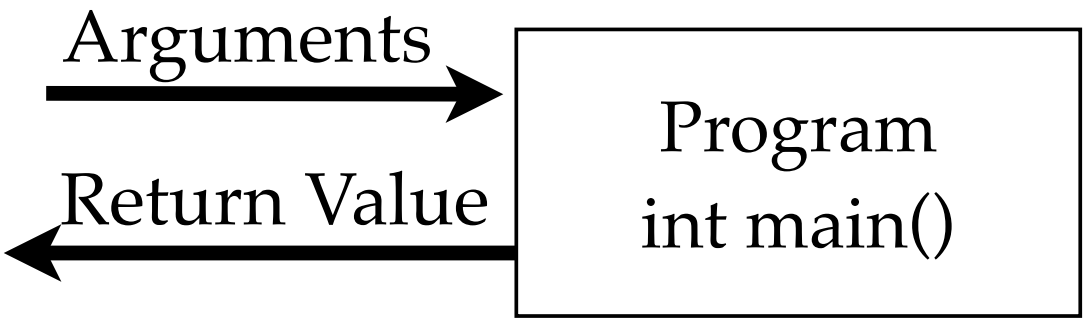

   return 0;
}
```

/* comments */

global declarations:
#include external files

main Function

The main() of hello.c



- No **arguments**.
- Returns an integer **variable**.

```
/*
Bert Huang. My first program.
*/



#include <stdio.h>



int main()
{

   printf("Hello, World!\n");



   return 0;
}
```

return “0” to OS:
“everything is OK”

C Statements

- One-line commands
- Always end in semicolon ;
- Examples:
  - call function: `printf("hello"); /* from stdio */`
  - declare **variable**: `int x;`
  - assign variable value: `x = 123+456;`

Variables

- Placeholders for values: just like in Algebra
- C variables have **types:**
  - **int –** integer valued (1, -23, 128, -999)
  - **char** – ASCII character (a, b, $, \n)
  - **float –** decimal fractional numbers (1.2, 0.3)
- Variables must be **declared**

Variables

- Declaring a variable:
  - I want to use a variable of this type...
  - I will refer to it as...
  - `int counter;`
  - `float ratio, ratio2;`
  - `char firstLetter, secondLetter;`

Manipulating Variables

- Basic arithmetic operators: `^ * / + -`
  - Obey order of operations
  - Use parentheses () to override order of operations.
  - `z = (x+y)/2;`

Types

- Why do we need types?
  - Different types are represented differently in memory.
  - Example: Can't efficiently represent fractional numbers in base-2.

int

- 4 bytes (on Unix)
- Base-2 representation.
- need one bit for + or -
- Range: -2^31 to 2^31
- Variants: short (2 bytes), long (8 bytes), unsigned (only non-negative)

char

- 1 byte
- ASCII representation in base-2
- Range: 0-255 (lots of unused)

float

- Stands for “floating decimal point”
- 4 bytes
- Similar to scientific notation: 4.288 * 10^3
- Very different interpretation of bits than int and char.
- Range: -10^(38) to 10^(38)

Casting

- We can **cast** a variable as a different type than its actual type:
  - ```
    float x;
    int y;
    y = 3;
    x = (float) y;
    ```
- Casting allows us to correctly use variables of different types together.
    ```

printf

- printf([formatted text], [arguments],...);
- Use placeholders for variables:

  `%d` int

  `%f` float

  `%c` char

- Examples:

```
printf("%d plus %d is %d\n", x, y, x+y);
```

Reading

- Practical C Programming: Chapters 3 and 4.