Introduction to C Programming

Stanley Yao
Computer Science Department
University of Arizona

Outline

- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

Background about C

- Originally developed by Brian Kernighan and Dennis Ritchie to write UNIX (1973)
- Intended for use by expert programmers to write complex systems
- Between the low level languages (e.g. assembly) and high level languages (e.g. BASIC)
- Powerful, flexible, efficient, but difficult to use, because you need to know every detail

Standardization & Productization

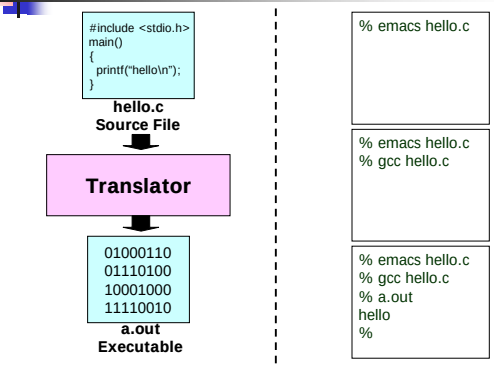
- 1st C standard was K&R, 1978
- Standardized by ANSI committee in 1989
- Extended features by vendors
  - Microsoft C
  - Borland C
  - POSIX Standards
  - Gnu C Library

- Next Step: C++ and OOP

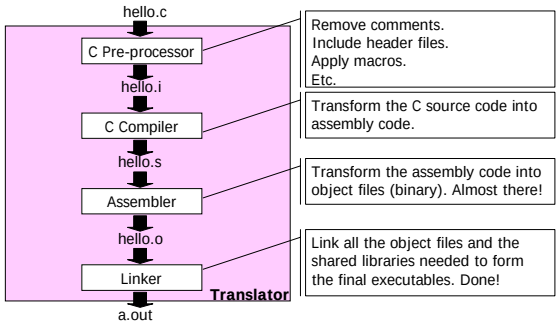
Outline

- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

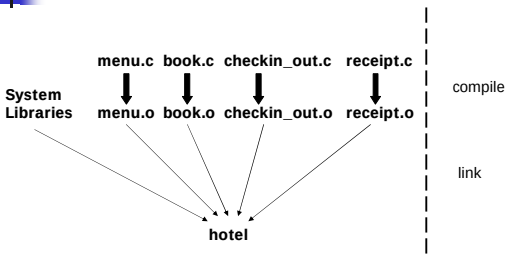
Developing C Programs



More Detail in the Translator



Larger C Project



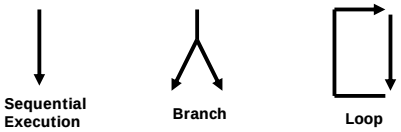
Outline

- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

Programs

- A way of communicating between human and computers.
- A description of what human want the computer to do.
- A list of commands onto the computer. Commands are not necessarily only executed in sequential order. (Control flow)

Basic Control Flow



Sequential Execution

Branch

e.g.
if…else…
switch

Loop

e.g.
for loop
while loop
do…while loop

Our First C Program

```
#include <stdio.h>
main()
{
    printf("hello, world\n");
}
```

- Written in a strict syntax; the compiler will check the syntax error
- A program consists of:
  - Global variables, and
  - Functions
    - Local variables, and
    - Statements: specify computing operations to be done
      - e.g. assignment, loop, function call, etc.

Syntax & Semantics

- Syntax: how the characters and words of a program must be put together. It is like the spelling and grammar of the programming language.
- Semantics: what the program means, i.e. what it does.
- “Syntactically correct” does not necessarily mean “semantically correct”
- Compiler will give messages on syntax errors
- Fixing semantic errors are far more difficult than fixing syntactic errors

Line-oriented C Code Format

- Line-oriented: newline = space
- Code 1:

```
#include <stdio.h> void main(void) { printf(“Hello
  World\n”);}
```

- Code 2:

```
#include <stdio.h>
void
main(
void) { printf(
“Hello World\n”
); }
```

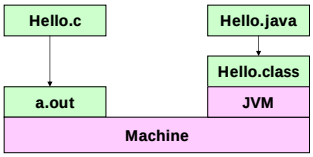
- Keep a good indentation style: make your life easier!
- Case sensitive

C .vs. Java

- C is procedure based; Java is object-oriented
- A C program is a collection of functions and global variables; Java wraps everything into objects
- C doesn’t have nested function definitions; Java allow nested class definitions
- C has pointers; Java has handles but disallow the direct pointer arithmetic

C .vs. Java (cont.)

- C programs are compiled into machine code; Java programs are compiled into byte code
- C programs need to be recompiled on different platforms; Java “compile once, run everywhere”



Variable

- Store the data that you operate on
- Different types: integer, character, pointer, etc.
- Variable Name
  - Made up of: letters (including “_”) and digits
  - Must begin with a letter
  - Can’t be C keywords
- Different scope:
  - Global variables: defined outside of functions
  - Local variables: defined inside of functions
- Declaration: `int a;`
- Assignment: `a = 10;`
- Reference: `b = a+1;`

Function

- Usually designed to accomplish a single job
- A function consists of:
  - Local variables to store data, and
  - Statements specifying the computing operations to be done
- Building blocks for a whole program
- A special function “main()”, all programs’ execution starts from main(). OS makes this happen.

Function (cont.)

- Function prototype: int mult2(int a);
- Function definition: <*see next slide*>
- Function call: b = mult2(a);

- Prototype helps the compile-time error checking
- "`#include <stdio.h>`" is a preprocessor directive, inserting the file stdio.h in the current place
- stdio.h has the prototypes of standard IO functions.
- Including stdio.h here is for function call checking, picking out the wrong calls during the compile time

Function Definition

```
int mult2(int a)
{
  int result;
  result = a + a;
  return result;
}
```

- Function header
  - Return type
  - Function name
  - Formal parameters
- Function body (between { and })
  - Variable declarations
  - Statements: usually end with ";"

Block

- Braces { and } and the group of declarations and statements inside the braces form a compound statement, or block.
- A block is syntactically equivalent to a single statement.
- Examples:
  - Function body
  - Blocks in the if, else, while and for statements

Outline

- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

Back to Our First C Program

```
#include <stdio.h>
main()
{
    printf("hello, world\n");
}
```

- Compile and run the program:

```
% gcc hello.c
% a.out
hello, world
%
```


#include <stdio.h>

- During the "Pre-processor" phase, this line will be replaced by the contents of the file stdio.h
- "stdio" means "Standard I/O"
- stdio.h has the declaration of functions like printf(), getc(), etc.

printf(“hello, world\n”);

- Print the desired information on the standard output (usually our terminal screen)
- A call to the function “printf” already built in the standard C library
- “printf” is the name of the function
- “hello, world\n” is the only parameter (of type character array) passed to the function
- #include <stdio.h> include the header file, in which the prototype of the function “printf” is specified (will talk about prototype in detail later)

printf()

- **Prototype**
  int printf(const char * *format*, ...);
- %d
- %f
- %c
- %s
- %%
- Specifying width: %3d

Escape Sequence

- Provide a general and extensible mechanism for representing hard-to-type or invisible characters. It is a SINGLE character.
- \n: newline character
- \t: tab
- \b: backspace
- \”: double quote
- \\: backslash itself

Outline

- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

More Examples 1 (printf)

```
/* Multiple printf’s give the same result */
#include <stdio.h>
main()
{
   printf(“hello, ”);
   printf(“world”);
   printf(“\n”);
}
```


More Examples 2 (loop, functions)

```
/* Calculate the n^i, where n is 2 and -3, i is from 0 to 9 */

#include <stdio.h>
int power(int m, int n);
main()
{
    int i;
    for (i=0; i<10; ++i)
        printf(“%d,%d,%d\n”, i, power(2, i), power(-3, i));
    return 0;
}
int power(int base, int n)
{
   int i, p=1;
   for (i=1; i<=n; ++i)
        p=p*base;
   return p;
}
```

More Examples 3 (nested loops)

```
/* Print a triangle */
#define MAX 10
#include <stdio.h>
main ()
{
  int i, j;
  for (i=1; i<MAX; i++) {
      for (j=0; j<MAX-i; j++)
            printf(“ ”);
      for (j=0; j<i; j++)
            printf(“*”);
      printf(“\n”);
  }
}
```


More Examples 4 (branch)

```
/* Pick the scores that are A’s and print them out */

#include <stdio.h>
main ()
{
  int i, score[]={96, 85, 100, 83, 35, 73};
  for (i=0; i<6; i++)
      if (score[i] > 90)
            printf(“%d --- A\n”, score[i]);
}
```