Introduction to C Programming

Alon Efrat

Computer Science Department
University of Arizona

Outline

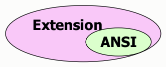
- Background to C
- How to develop a C program
- Basic Concepts
- Study Our First C Program
- Tasting Some Sample C Programs

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.

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

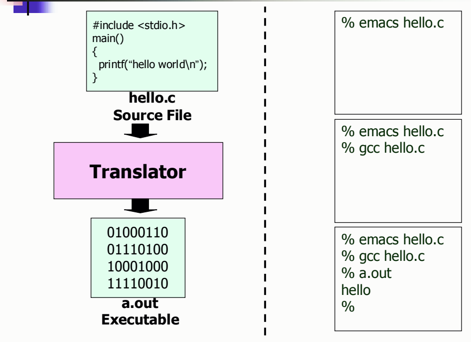


- Next Step: C++ and OOP

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



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

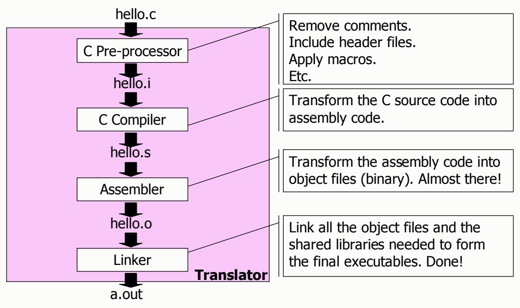
```
01000110
01110100
10001000
11110010
```

```
% emacs hello.c
```

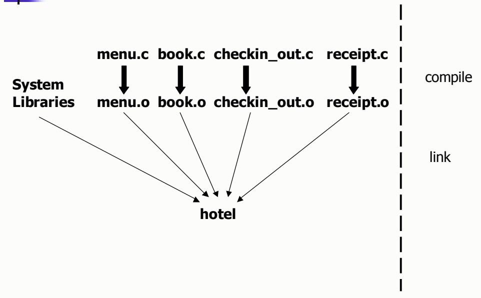
```
% emacs hello.c
% gcc hello.c
```

```
% emacs hello.c
% gcc hello.c
% a.out
hello
%
```

More Detail in the Translator


hello.c
C Pre-processor
hello.i
C Compiler
hello.s
Assembler
hello.o
Linker
Translator
a.out
Remove comments. Include header files. Apply macros. Etc.
Transform the C source code into assembly code.
Transform the assembly code into object files (binary). Almost there!
Link all the object files and the shared libraries needed to form the final executables. Done!

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

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"
); }
```

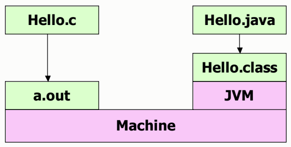
- Keep a good indentation style: make your life easier!

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
  - External variables: known to different modules
- 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.)

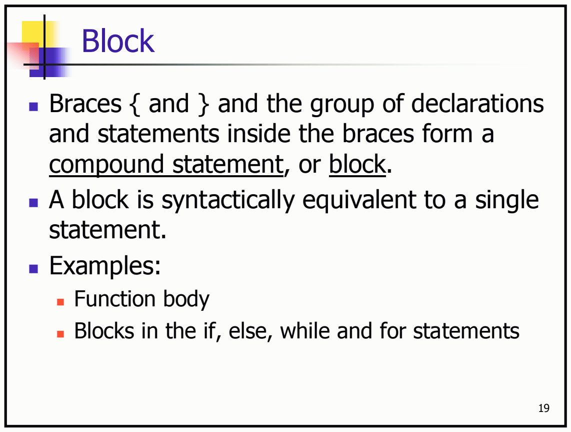
- Function prototype: int mult2(int a);
  Specifying the existence of function,
  can appear many times.
- Function definition: <*see next slide*>
- Function call: b = mult2(a);

- 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;
}
```

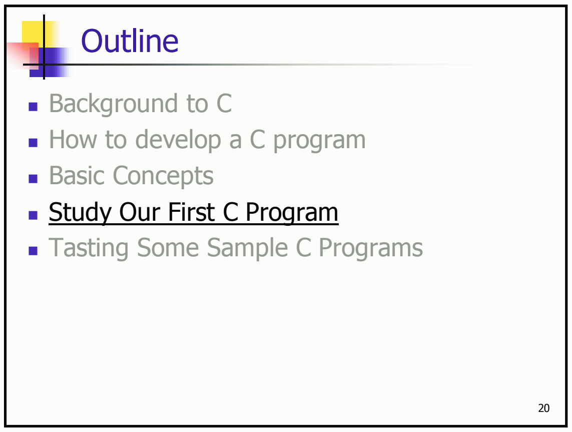
- •Function header
  - •Return type
  - •Function name
  - •Formal parameters
- •Function body (between { and })
  - •Variable declarations
  - •Statements: usually end with ";"



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

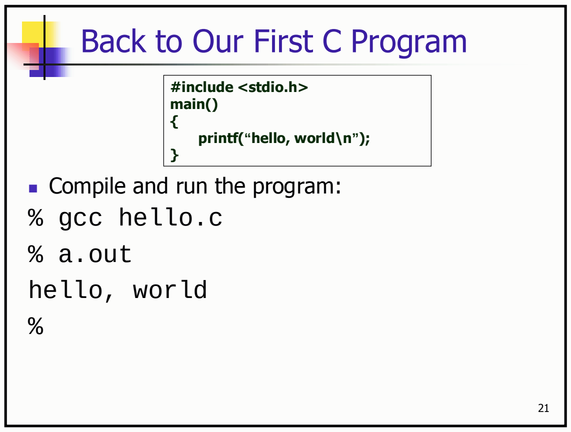
19



Outline

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

20



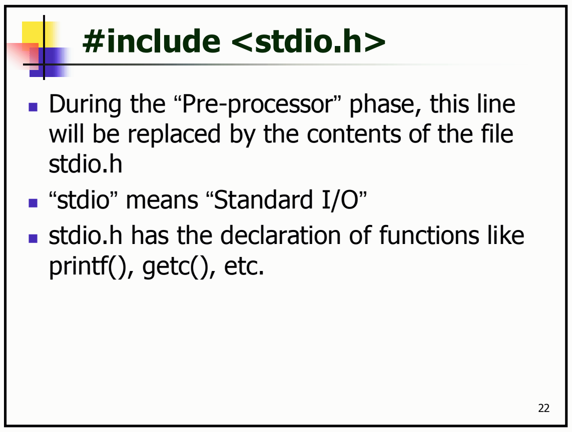
Back to Our First C Program

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

- Compile and run the program:

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

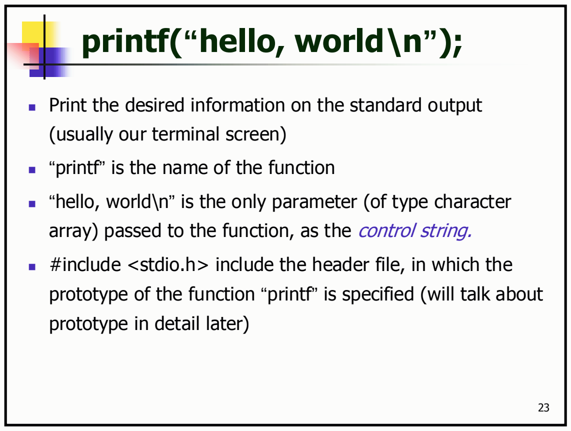
21



#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.

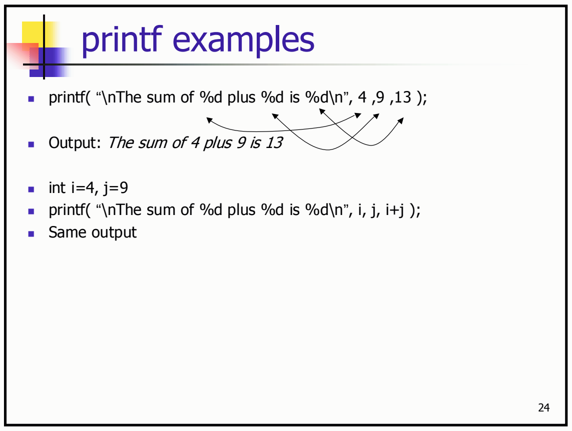
22



printf("hello, world\n");

- Print the desired information on the standard output (usually our terminal screen)
- "printf" is the name of the function
- "hello, world\n" is the only parameter (of type character array) passed to the function, as the *control string.*
- #include <stdio.h> include the header file, in which the prototype of the function "printf" is specified (will talk about prototype in detail later)

23



printf examples

- printf("\nThe sum of %d plus %d is %d\n", 4 ,9 ,13);
- Output: *The sum of 4 plus 9 is 13*

- int i=4, j=9
- printf("\nThe sum of %d plus %d is %d\n", i, j, i+j);
- Same output

24

printf()

- %d - decimal number (e.g. 17)
- %f - floating point (e.g. 3.1415)
- %c - single char (e.g. H)
- %s - a string (unlimited length, e.g. “hello world”
- %% - the character %
- Specifying width: %3d, %6.2f

25

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

26

Outline

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

27

More Examples 1 (printf)

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

28

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);      /* Prototype */

main()
{
    int i;
    for (i=0; i<10; ++i)      /* loop – run from i=0 to i=10 */
        printf(“%5d,%5d,%7d\n”, i, power(2, i), power(-3, i));
    return 0;
}

int power(int base, int n) /* calculate base^n */
{
   int i, p=1; /* note that I is a local variable */
   for (i=1; i<=n; ++i)       /* runs
        p=p*base;
   return p;
}
```

29

Output of this program

```
    0,    1,      1
    1,    2,     -3
    2,    4,      9
    3,    8,    -27
    4,   16,     81
    5,   32,   -243
    6,   64,    729
    7,  128,  -2187
    8,  256,   6561
    9,  512, -19683
```

30

More Examples 3 (nested loops)

```
/* Calculate the multiplication table */
#define MAX 10
#include <stdio.h>
main () {
   int i, j;
   for (i=1; i<MAX; i++) {
      for (j=0; j<MAX-i; j++)
         printf("|%4d",i*j);
      for (j=0; j<i; j++)
         printf("     ");
      printf("\n");
   }
}
```

Output of ex3

```
▪ |   1|   2|   3|   4|   5|   6|   7|   8
▪ |   2|   4|   6|   8|  10|  12|  14
▪ |   3|   6|   9|  12|  15|  18
▪ |   4|   8|  12|  16|  20
▪ |   5|  10|  15|  20
▪ |   6|  12|  18
▪ |   7|  14
▪ |   8
```

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};
      /*An array initialized to contain these numbers*/
      /*The size of the array is determined by the num
      of elements */
  for (i=0; i<6; i++)
      if (score[i] > 90)
            printf(“%d --- A\n”, score[i]);
}
```