Topic 1. Simple C Programs

COMP ENG 2SH4

Principles of Programming

McMaster University, 2015

Instructor: Sorina Dumitrescu

C Programming Language

- C is a language for procedural (imperative) programming
- C is a **compiled** language:
  - A compiler translates the C program into machine language (the language that is directly ”understood” by the computer).
- It was developed in the early 70’s by Dennis Ritchie. (It evolved from two previous languages).
- It was used to develop UNIX.
- Many of today’s leading operating systems are written in C or C++.

C Standards

- C expanded rapidly over various hardware platforms leading to many variations.
- C89: approved by ANSI in 1989, by ISO in 1990,
- Other standards:
  – C99 - ISO/IEC 9899-1999
  – C11 (ISO/IEC 9899-2011)
  – not supported by all compilers
- C89 was adopted by almost all compilers. Most existing C code is compliant to C89.
- **We will study C89 mostly.**

C Standard Library

- The C Standard Library is
  - a collection of existing functions providing functionality for I/O, string manipulation, simple math calculations, etc.

Simple Program in C. Parallel with Python

- Printing a line of text

```
# Python code
def main():
     print “Principles of Programming”
```

```
/* C code */
#include <stdio.h>

int main(void)
{
    printf(“Principles of Programming\n”);
    return 0;  /* indicates that program ended
                      successfully */
}
```

Simple Program in C

- Every C program contains the function **main**()
- Program execution starts at **main**
- Keyword **int** in front of main() indicates that the function returns a variable of type **int** (integer).
- The statements to be executed by **main**() are enclosed within **{ }**. They form the body of the function.
- **int main(void)** is the header of the function

Simple Program in C

- To print text on the screen, use printf() (function defined in the C standard library).
- Format: printf("text to be printed");
- When using I/O functions: **#include** **<stdio.h>** (this is a preprocessor directive)
- Notice the **semicolon** at the end of every statement.

Simple Program in C

```
// printing two lines of text
#include <stdio.h>

int main(void)
{
    printf(“Principles of Programming\n”);
    printf(“COMP ENG 2SH4\n”);
    return 0;
}
```

- \n is an escape sequence. It indicates **newline**.
- Multi-line comments: enclosed within /* */
- Single line comments: after //

Simple Program in C. Parallel with Python

- Read an integer and print its squared value.

```
/* in Python */
def main
   num = input(“Enter an integer: ”)
    print “The square of “, num, “ is “, num*num
```

```
/* in C */
#include <stdio.h>
int main(void)
{   int num; /* declaration of variable num */
    printf(“Enter an integer: ”);
   scanf(“%d”, &num); /* read the integer and
               store the value in variable num */
    printf(“The square of %d is %d\n”,num,num*num);
    return 0;
}
```

Simple Program in C

- Every variable has to be **declared** first with a type.
- All declarations have to appear before the executable statements.
- In general a variable declaration is also a **definition,** i.e. it **allocates memory for the variable.** The type specifies the amount of memory, what kind of values it can take, etc.

Explanation of scanf()

- To input from the standard input stream (usually, keyboard) use function **scanf**( )
- Explanation of: `scanf(“%d”, &num)`
- “%d” in scanf() is a **conversion specification**: scanf() reads the sequence of characters input at the keyboard until the first empty space and converts it to an integer (so it has to be a sequence of digits possibly with a sign at the beginning)
- **&num** in the call to scanf() indicates the memory location where variable **num** is stored
- The value read by scanf() will be stored at that location (i.e., it is assigned to variable **num**)

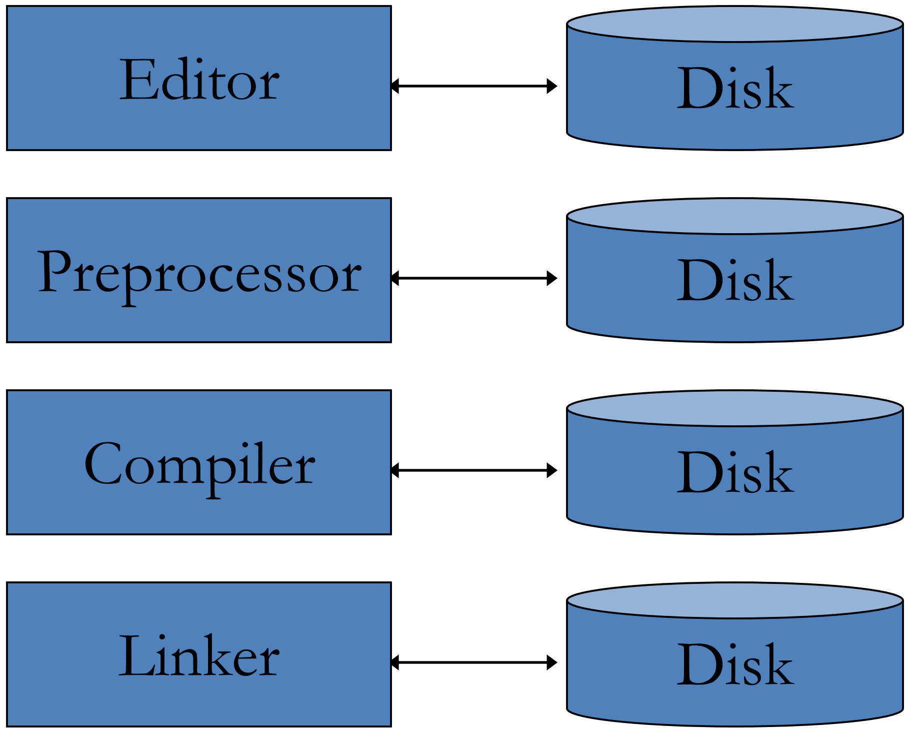
Conversion Specifiers

- printf() and scanf() use conversion specifiers
- int: %d
- char: %c
- float: %f
- double: %f (printf);   %lf (scanf)

Explanation of printf()

- Explanation of
  `printf(“The square of %d is %d\n”,num,num*num)`
- The text within “ “ is printed on the screen character by character, except for the conversion specifiers.
- In place of each conversion specifier, the corresponding value from the list after the text is printed.
- Value of num is printed in place of first %d, value of num*num is printed in place of second %d.

C Development Environment



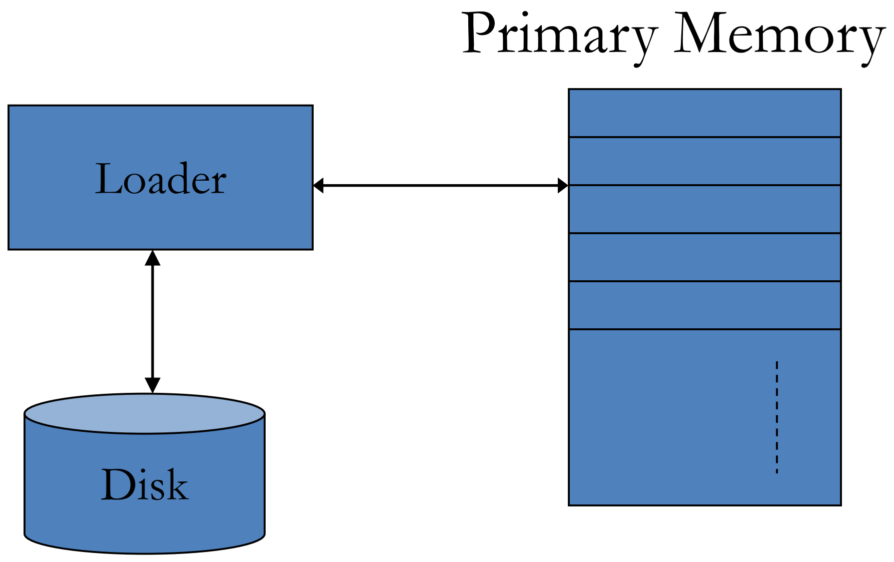
Program is created in the editor and stored on disk.

Preprocessor program processes the code.

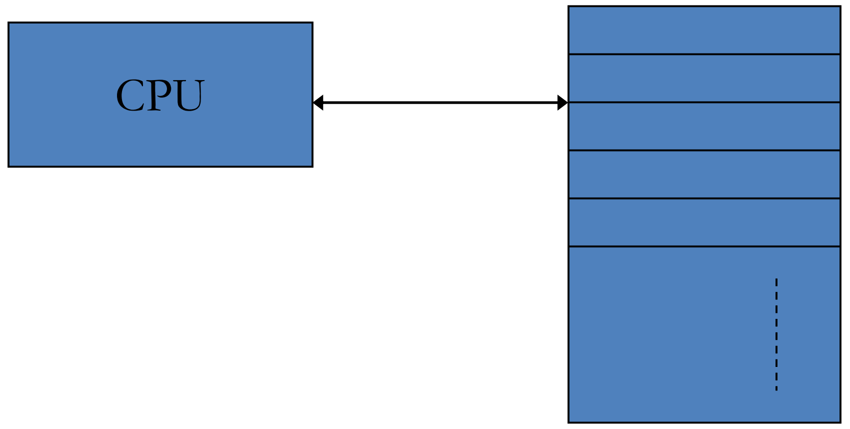
Compiler creates object code and stores it on disk.

Linker links the object code with the libraries

Running Environment



Loader puts program in memory.



CPU takes each instruction and executes it, possibly storing new data values as the program executes.

C Development Environment

- Editing:
  - Writing the C code in a text editor
- Preprocessing
  - Automatically executed at the ”compile” command.
  - Certain manipulations are performed on the program as indicated by the **preprocessor directives**. Ex:
    - including other files in the file to be compiled (indicated by #include directive). Ex: #include < stdio.h >
    - performing various text replacements. Ex: #define SIZE 10 (SIZE will be replaced by 10 allover the program)

C Development Environment

- Compilation:
  - Translation into machine language code → **object code**.
- Linking
  - The object code is linked with the code for the missing functions (i.e., functions referred to in the program, but defined elsewhere) → **executable image** (.exe on Windows)