CC-112 Programming Fundamentals

The C Programming Language

Nazar Khan
Department of Computer Science
University of the Punjab

The C Programming Language

- Evolved from B programming language by Dennis Ritchie at Bell Laboratories.
- Originally implemented in 1972.
- Development language of the UNIX operating system.
- Many of today's leading operating systems are written in C and/or its successor C++.
- C is mostly hardware independent – the same C program can run on different computers.

Built for Performance

- C is widely used to develop systems that demand performance, such as
  - operating systems
    - Linux,
    - portions of Windows and Android,
    - Apple's OS X written in Objective C which is a derivative of C
  - embedded systems
    - run fast
    - conserve power
    - conserve memory
  - real-time systems
    - for *mission-critical* applications
    - 24/7, immediate, predicatble response
  - communications systems
    - massive amounts of data
    - sent to huge number of destinations
    - receiver's exprience should be smooth

Standardization

- As C became popular, versions for different computers (hardware platforms) were developed.
- A *standardized* version was *agreed upon* in 1989. Goal was to develop a machine-independent definition of the C language.
- Two more standardized version were agreed upon in 1999 (C99) and 2011 (C11).
- C11 is a refined and expanded version of traditional C.

Popular C-based languages

- C++
- Objective-C
- Java, JavaScript
- C#
- PHP
- Python
- Swift

C Standard Library

- C programs consist of pieces called functions.
- C provides a rich collection of existing functions called the *C Standard Library*.
- Thus, there are really two parts to learning how to program in C
  - learning the C language itself, and
  - learning how to use the functions in the C Standard Library.
- Avoid "reinventing the wheel". Use existing pieces – this is called *software reuse*.
  - C Standard Library functions
  - Functions you create yourself
  - Functions other people (whom you trust) have created and made available to you

C Standard Library

- The advantage of creating your own functions is that you'll know exactly how they work. You'll be able to examine the C code.
- The disadvantage is the time-consuming effort that goes into designing, developing, debugging and performance-tuning new functions.

Compiling and executing a C program in Windows

1. Install MinGW from `https://osdn.net/projects/mingw/downloads/68260/mingw-get-setup.exe/`
2. Add C:/MinGW/bin to the system variable called "Path"
3. Save the following code in a file called hello_world.c

```c
#include <stdio.h>

int main(){
    printf("Hello World!");
    return 0;
}
```

4. Open command prompt
5. Change directory to the folder where you saved hello_world.c
6. Enter the command *gcc hello_world.c*. This will create a file called *a.exe*.

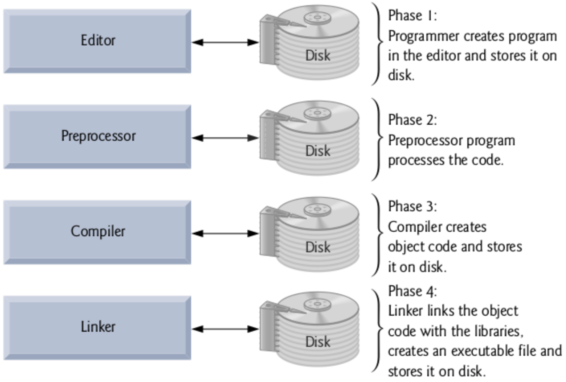
Compiling and executing a C program in Windows

7. Enter the command a.exe. This will print "Hello World!" on the screen.
8. Enter the command *gcc hello_world.c -o hello_world.exe*. This will create a file called *hello_world.exe* instead of *a.exe*. Run hello_world.exe and it will print "Hello World!" on the screen.

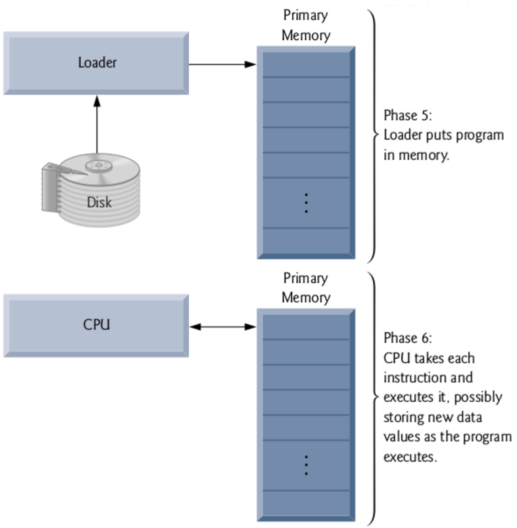
Typical C Program-Development Environment

- C systems generally consist of several parts:
  - a program-development environment
  - the language, and
  - the C Standard Library
- To run/execute a C program, it has to go through 6 phases
  1. edit and save with .c extension
  2. preprocess
  3. compile
  4. link
  5. load
  6. execute

Typical C Program-Development Environment


Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk.
Preprocessor
Disk
Phase 2:
Preprocessor program processes the code.
Compiler
Disk
Phase 3:
Compiler creates object code and stores it on disk.
Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.

Typical C Program-Development Environment


Loader
Disk
Primary Memory
Phase 5:
Loader puts program in memory.
CPU
Primary Memory
Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.