Introduction to Programming I

Alexey Shikulin, Munir Makhmutov, Sami Sellami and Furqan Haider

C programing language

- C is a programming language invented by Dennis Ritchie that first appeared in 1972.
- It's what we call a low-level language, meaning there is only little abstraction between C and machine language, so it can be considered to be closer to the computer's hardware.
- C is also a **compiled** language, as opposed to interpreted, meaning the source files written in C should be compiled in order for them to be executable.

Tools

You only need:

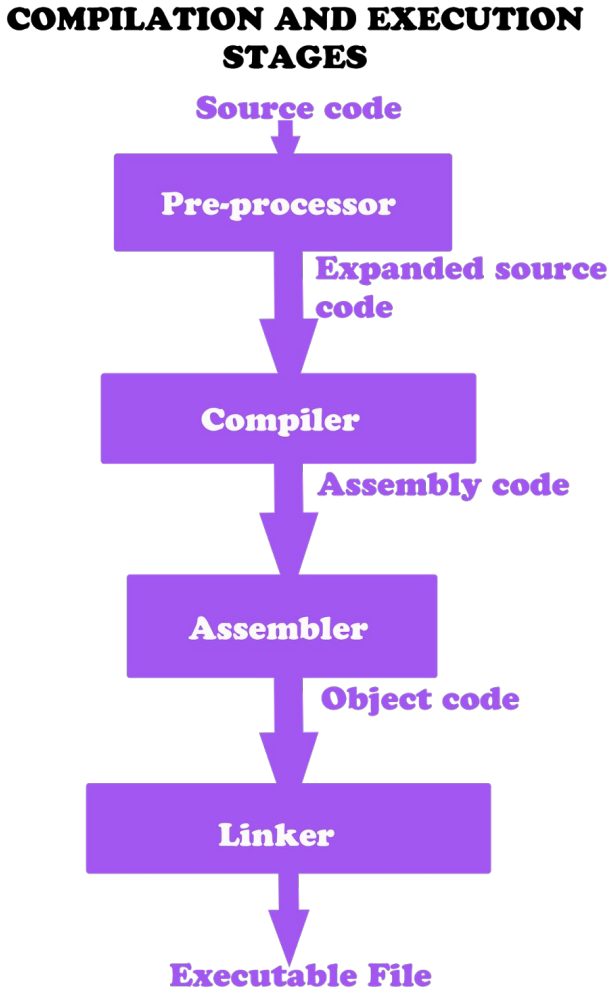
(a) Text Editor

(b) The C Compiler.

memegenerator.net

Compilation:

Compilation is the translation of source code (the code we write) into object code (sequence of statements in machine language) by a compiler.



Installation of gcc - on UNIX/Linux.

```
$ gcc -v
```

If you have GNU compiler installed on your machine, then it should print a message as follows −

```
Using built-in specs.
Target: i386-redhat-linux
Configured with: ../configure --prefix=/usr .......
Thread model: posix
gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
```

If GCC is not installed, then you will have to install it yourself using the detailed instructions available at https://gcc.gnu.org/install/

Installation of gcc - on MacOS.

If you use Mac OS X, the easiest way to obtain GCC is to download the Xcode development environment from Apple's web site and follow the simple installation instructions. Once you have Xcode setup, you will be able to use GNU compiler for C/C++.

Xcode is currently available at developer.apple.com/technologies/tools/.

Installation of gcc - on Windows.

To install GCC on Windows, you need to install MinGW. To install MinGW, go to the MinGW homepage, www.mingw.org, and follow the link to the MinGW download page. Download the latest version of the MinGW installation program, which should be named MinGW-<version>.exe.

Linux subsystem for **Windows**

More about gcc

- Run **gcc**:

  gcc -Wall infilename.c -o outfilename.o

- ***-Wall*** enables most compiler warnings
- More complicated forms exist
  - multiple source files
  - auxiliary directories
  - optimization, linking
- Embed debugging info and disable optimization:

  gcc -g -O0 -Wall infilename.c -o outfilename.o

More about gcc. Compiling Simple Programs

Say you have a file **hello.c** as follows :

```c
#include <stdio.h>

int main() {
   /* my first program in C */
   printf("Hello, World! \n");

   return 0;
}
```

More about gcc. Compiling Simple Programs

You can compile and run it from the unix prompt as follows :

gcc hello.c

This creates an **executable** called "a.out". You can run it by typing

./a.out

at the prompt. a.out is the default name for an executable. Use the "-o" option to change the name :

gcc hello.c -o hello

creates an executable called "hello". You can run it by typing

./hello

Debugging

- All program to be debugged in gdb must be compiled by gcc with the option "-g" turning on. We can simply start gdb by:

  gdb ./hello

Figure: gdb: command-line debugger

Debugging

Some useful commands:

- ***break linenumber*** – create breakpoint at specified line
- **break file:linenumber** – create breakpoint at line in file
- **run** – run program
- **c** – continue execution
- **next** – execute next line
- **step** – execute next line or step into function
- **quit** – quit gdb
- **print expression** – print current value of the specified expression
- **help command** – in-program help

More about gcc. Include Directories

Sometimes the header files that you write are not in the same directory as the .c file that #include's it. E.g. you have a structure in a file *foo.h* that resides in */homes/me/include*. If you want to include that file in *hello.c* you can do the following:

- add #include <foo.h> to hello.c
- compile with the -I option:

  gcc -o hello hello.c -I /homes/me/include

This basically tells gcc to look for #include's in /homes/me/include **in addition** to other directories you specify with -I

Structure of a .c file

/* Begin with comments about file contents */

Insert ***#include*** statements and preprocessor definitions

Function prototypes and variable declarations

Define main() function {

    Function body

}

Define other function

{

    Function body

}

Comments

- Comments: /* this is a simple comment */
- Can span multiple lines

```
/* This comment spans
multiple lines */
```

- Completely ignored by compiler
- Can appear almost anywhere

The #include macro

- Header files: constants, functions, other declarations
- #include <stdio.h> – read the contents of the header file stdio.h
- stdio.h: standard I/O functions for console, files

```
#include <stdio .h> /* basic I /O facilities */
```

More about #include

- **stdio.**h – part of the C Standard Library
- Other important header files: **ctype.h, math.h, stdlib.h, string.h, time.h**. For the ugly details visit www.unix.org/single_unix_specification/ (registration required)
- Included files must be on include path
- **-I directory** with **gcc**: specify additional include directories
- standard include directories assumed by default
- #include "stdio.h" – searches ./ for stdio.h first

Task 1: Write Hello World Example

```c
#include <stdio.h>

int main() {
   /* my first program in C */
   printf("Hello, World! \n");

   return 0;
}
```

Executing Hello World Example

- Open a text editor and add the above-mentioned code.
- Save the file as *hello.c*
- Open a command prompt and go to the directory where you have saved the file.
- Type *gcc hello.c* and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate *a.out* executable file.
- Now, type *a.out* to execute your program.
- You will see the output *"Hello World"* printed on the screen.

Declaring Variables

- Must declare variables before use
- Variable declaration:
  - ***int n;***
  - ***float phi;***
- **int** - integer data type
- **float** - floating-point data type
- Many other types (more next lecture. . .)

Initializing Variables

- Uninitialized variables assume a default value
- Variables initialized via assignement operator:

  n = 3;

- Can also initialize at declaration:

  float phi = 1.6180339887;

- Can declare/initialize multiple variables at once: int a, b, c = 0, d = 4;

Arithmetic Expressions

- Suppose ***x*** and ***y*** are variables

  x+y, x-y, x*y, x/y, x%y: binary arithmetic

- A simple statement: ***y = x+3*x/(y−4);***
- Numeric literals like **3** or **4** valid in expressions
- Semicolon ends statement (not newline)
- ***x += y, x -= y, x *= y, x /= y, x %= y:*** arithmetic and assignment

Task 2

Write a program which declares/initializes 2 integer variables and prints the result of their addition, subtraction, multiplication and division.

X, Y

X + Y

X - Y

X * Y

X / Y

References

1. https://www.geeksforgeeks.org/interesting-facts-about-c-language/
2. https://www.improgrammer.net/interesting-facts-c-language/
3. https://www.tutorialspoint.com/cprogramming/index.htm
4. https://courses.cs.washington.edu/courses/cse451/98sp/Section/gccintro.html
5. http://hilite.me/