History

C is a general purpose programming language. It was developed by Dennis Ritchie at AT&T Bell Laboratories in the year 1972. It was developed along with the Unix Operating System. C was extensively used in its development. It was first implemented on the Digital Equipment Corporation PDP-11 computer.

C is an outgrowth of an earlier language B which was itself based on BCPL (Basic Combined Programming Language).

The Unix operating system and almost all Unix applications are written in the C language. C has now become a widely used professional language for the following reasons:

- It produces efficient programs.
- It can handle low-level activities.
- It can be compiled on a variety of computers.

The C language was formalized in 1988 by the American National Standard Institute (ANSI). The latest standard is referred to as c11 and stands for the ISO/IEC 9899:2011. Since that is relatively new, the most often used standard is likely to be c99. We will be using c99 in this course.

C usage

C was initially used for system development work. It is suitable for this, as it produces code that runs nearly as fast as assembly.

Some such uses of C are:

- Operating Systems: Unix, Linux
- Language Compilers: CCS C Compiler, Dev-C++, Digital Mars
- Assemblers
- Text Editors: vi Text Editor
- Network Drivers
- Language Interpreters: Python

Sample Program

Now let us write our first C program. It will display *Hello World!* on the screen. To create, compile and execute the program, follow these steps:

1. Open the file using **vim Program-01-1.c** and type the following lines in that file and save.

```
1 /* My first program */
2 #include <stdio.h>
3 int main() {
4     printf("Hello World!\n");
5 }
```

2. Now type the below command in terminal to compile the program.

```
c99 Program-01-1.c
```

3. Now type the below command in terminal to execute the program.

```
./a.out
```

Whenever the source program is modified, the entire process of compiling and executing the program should be repeated.

> **Note**
> In Unix/Linux, by default the C compiler will produce an executable file called `a.out`. We can either give another name while compiling or rename the executable file after compiling. For example, `c99 Program-01-1.c -o hello` will produce an executable named `hello`.

Toolset

Three things are necessary for creating C programs:

Text Editor A text editor is a program used for writing and editing plain text. It differs from a word processor in that it does not do formatting such as typefaces, fonts, margins and italics. C programs can be written using any of the many text editors that are available for Linux such as `vim, gedit, nano, emacs`.


```
$ vim Program-01-1.c
$ c99 Program-01-1.c
$ ./a.out
Hello World!
$
```

Figure 1: Output

Compiler A compiler is a specialized program that converts source code into machine language (also called object code or machine code) so that it can be executed by a CPU (central processing unit).

C Standard Library The C standard library is a collection of header files and library routines used to implement common operations, such as input/output and string handling, so nearly all C programs rely on the standard library functions.

There are four phases involved in compilation process:

Preprocessing The C preprocessor –in Unix/Linux it is a program called cpp– is a macro processor that is used automatically by the C compiler to transform your program before compilation. It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.

Compilation Source code for C programs are written as ASCII text. They must be *compiled* and *linked* to produce *executable* programs. The executable files are what you actually run. The compilation of code involves several steps:

- parsing of the statements for syntax
- translation of the statements into assembly language instructions
- setting up the addresses of all variables and functions

Assembly C is a high-level compiled language. The process of compilation is very complex, but the basic operation resembles that of assembly, with one exception – each line of the program can produce one or more machine code instrcutions. The assembler takes

as its source code an assembly language program, this is also a file of ASCII characters, it used this to produce machine code.

Linking When a programmer needs a library, he specifies the library to the compiler/assembler. The compiler/assembler then doesn't try to find the address of the procedure, but instead leaves this to the linker. It provides the name of the appropriate libary file to the linker. The linker then processes the object code which was generated. It searches the known sets of library files and connects up the references to the library code with the actual code.

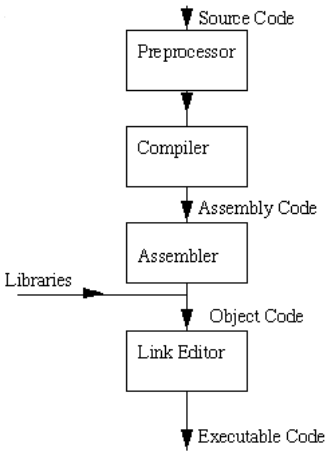

Figure 2: Compilation Phases

Building blocks of C Program

In order to understand the functioning a program, we need to understand the role of the following four elements: Preprocessor commands, Functions, Variables and Comments. It should be noted that this perspective is in no way intended as describing the structure of a program.

Preprocessor Commands All preprocessor lines begin with '#'. These commands tells the compiler to do preprocessing before doing actual compilation. For example, `#include <stdio.h>` is a preprocessor command which tells a C compiler to include `<stdio.h>` file before going to actual compilation. The C Preprocessor is not part of the compiler, but it is a separate program called internally by the compiler.

Functions Functions are main building blocks of any C Program. Every C Program will have one or more functions and there is one mandatory function which is called `main()` function. The C program execution starts from *main()* function. We will discuss functions in detail later.

Variables Variables are simply names used to refer to some location in memory, a location that holds a value with which we are working. Within the C programming language, when managing and working with variables, it is important to know the type of variables and the size (memory required) of these types.

Statements Statements can be expressions, assignments, function calls, or control flow statements which make up C program. Expressions combine variables and constants to create new values.

Block A block is a lexical grouping of statements and can be thought of as *one* logical statement. A block in C is delimited by braces: { and }.

Comments Comments are used to give additional useful information inside a program. you can place comments in your code that are not executed as part of the program. A block comment is used when the comment will span several lines. A block comment starts with /* symbol and ends with */ and can be anywhere in your program. The line comment uses two slashes (//), generally used for short comments.

Header Files

All header files should be included explicitly before `main()` function. They have the extension *.h*.

A header file contains declarations of different predefined functions, which are used in the program. A header file may also contain macros. This allows programmers to separate functions of a program into reusable chunks.

The prototypes of library functions are gathered together into various categories and stored in header files. For example, all prototypes of standard input/output functions are stored in header file *'stdio.h'*, while the mathematical functions such as sine are in *'math.h'*.

The header files can be declared in two ways.

```
#include "Aheader"
#include <Bheader>
```


Now Aheader will be looked for in the current directory, while Bheader will be looked for in the standard header file locations: most probably '/usr/include' in Unix systems.

Standard Headers

Following are the most commonly used header files, presented in alphabetical order.

- assert.h
- ctype.h
- graphics.h
- math.h
- process.h
- stdio.h
- stdlib.h
- string.h
- time.h

Naming Rules in C

There are rules for choosing the character sequence to be used for identifiers which denote variables, types and functions. Such identifiers must begin with an alphabet or underscore, and may be followed by any combination of alphabets, underscores, or the digits 0-9.

Please note that C is case sensitive; uppercase and lowercase alphabets are treated as distinct.

You should ensure that you use meaningful names for your identifiers. Please note meaningful *does not* necessarily mean long. The goal is to make the program easier to read and be self-documenting. Compare the two segments:

```
distance = speed * time;
s = v * t;
```

Most users adopt the convention that variable names are all lower case, while names for constants are all upper case.

Keywords

Keywords are reserved identifiers that have strict meaning to the C compiler and hence cannot be used as identifiers in your program.

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Table 1: List of C keywords

Additional notes

Note

C is a procedural language. A procedural language is a programming language that uses procedures (aka subroutines or functions) as the unit of execution and modularisation. A procedure contains a self-contained set of statements that execute to produce a desirable output.

