classmate
Date 16-6-20
Page

C - Programming Language

Session 1

→ Why C ?
→ Explanation of plateforms (PC, mobile, OS, compiler, editor)
→ Basic commands of Linux
→ VI editor
→ Simple program demo

Program:- Program is a set of instruction. Every program is responsible for processing the data/performing some task.

Language:- Language is nothing but it is just how you are forming / writting these instructions.

Embedded system:-
- The system which is the combination of software and hardware to perform multiple task.
- A small system where a combination of hardware, some software are their which is designed for some task / which can perform multiple task.
- Embedded means write a program & dump it into hardware.
- A small system installed into a big system.
- Combination of software & hardware to perform multiple and single task.

C language :-

- C is a procedural programming language.
- It was initially developed by Dennis Ritchie between 1969 and 1973.
- C is a portable language.
- Low level access to memory.
- It is a middle level language.
- It can be used to write OS as well as application level programming.
- Simple set of keywords.
- Clean style

Advantage of C programming :-

- C is a Middle-level language.
  - The middle-level language are somewhere between the low-level machine understandable language and High-level user friendly language.
  - C reduces the gap between the low-level and high-level language.
  - It can be used for writing operating systems as well as doing application level programming.
- Helps to understand the fundamentals of Computer Theories.
  - In the modern high level languages, the machine level details are hidden from the user, so in order to work with CPU cache, memory, network adapters, learning C programming is a must.

- Fewer Libraries
  - C programming language has fewer libraries in comparison with other high-level languages.
  - So, learning C programming also clears programming concepts to a great extent as you have to write lot of things from scratch
- C is very fast in terms of execution time.
  - program written and compiled in C executes much faster than compared to any other programming language.
  - C programming language is very fast in terms of execution as it does not have any additional processing overheads such as garbage collection or preventing memory leaks etc.
  - The programmer must take care of these things on his own.
- Embedded programming.
  - Embedded programming is also referred to as micro-controller programming, where C program is used to control microcontroller.
  - Microcontroller and embedded programming is widely used in auto-motives, Robotics, Hardware's etc.

→ Hardware doesn't Know C language. We are writing the program & compiler converts the our program in the hardware understandable language i.e binary (1 & 0).

→ Compiler responsibility is to convert the code into machine understandable language.

Linux OS :-

- Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel.
- It initially released by Linus Torvalds on 1991.
- It is a free and open-source OS.
- The source code can be modified and distributed to anyone commercially or non commercially under the GNU General Public License.
- High Security, Large Community Support, High Stability and Higher performance

Embedded Linux :-

- Embedded Linux is a type of Linux operating system / Kernel that is designed to be installed and used within embedded devices and appliances like,
  - Consumer electronics (set-top boxes, smart TVs, Mobile phones)
  - In-Vehicle infotainment (IVI)
  - Networking equipment (such as routers, switches, wireless access points or wireless routers)
- Example: Android OS,

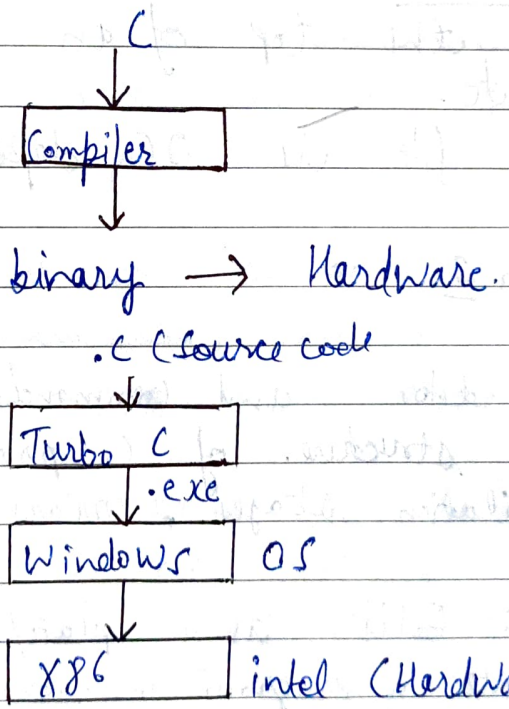
Advantages of Linux in Embedded Systems :-

- Easy customization.
- Used in device-specific purpose-built applications.

- Power consumption is low.
- Easily portable.
- Large community support
- Performance optimization
- Low cost.

Session 2:-

→ Native vs Cross compiler
→ Explanation of plateforms (PC, mobile, OS, editer), simple program demo
→ Basic structure of C program.
→ Compilation stages, errors.



Native compiler:-
The compiler which is working in one environment & generating machine understandable code for the same environment.
Ex:- Turbo C, GCC, DOC compiler etc.
Dev. C

Cross compiler:-

Cross compiler is a compiler which is working in one environment & generating the codes for other environment.
Ex:- Keil, WinAVR, Arm gcc etc.

→ Keil input as .asm (assembly file)
Keil output as .Hex file

Raw code:- Raw code is a code which can run/execute without an operating system.

Executatable (ex) code:- Executable code is a code on the top of an OS it will execute.
→ .exe file is OS dependent

Session 3:

→ Vi editor and commands
→ Basic structure of C program
→ compilation stages, errors

Editer:- Editer is a plateform where we write a program & transfer that program to the compiler.
It is basically like a piece of paper in a system/laptop where we are going to write a program/ create a source code & supply to compiler.
→ Some of the compiler have inbuilt compiler like turbo c, Keil. etc.
→ we are going to take help of vi-editer

Vi editor:-

- Vi editor works in 3 modes
  1. Insert mode → Press Insert button or i.
  2. Command mode → Vi editor by default in command mode, if we want to back from insert mode to command mode, just press escape button.
  3. ex (execute mode) → Press Escape + shift : or Escape + shift + zz

How to come out from Vi editor?

→ Press escape
→ Press shift + :
→ you need to type wq (wq is nothing it is just save and quite)
→ After that press enter button.

→ If we want to come out from vi editor without save any changes then press escape and shift + : ; type !q!

* → Compilers are extension dependent.
* → Or may or may not be.
* → In linux without extension program can be execute.
* → whenever you did any changes in source code you need to compile again or recompile.
* → When you press escape + shift :, your vi editor is in ex (execute) mode.

Linux basic commands

- **Commands:-**

→ ls : display list of files

→ Ctrl + l : to clear screen or clear screen

→ mkdir : to create directory

→ pwd (present working directory) : to show in which directory command prompt is working.

→ cd : change directory

- **Paths:**

→ Absolute path: starts with (/)

→ Referance / Relative path: starts with (. or ..)
Here (.) represents present working directory
(..) represents parent directory

- **Commands :-**

→ rmdir : to remove directory (if directory is empty)

→ rm -r : to remove directory (if directory is full)

→ vi filename.c : to create file

→ Esc + shift : wq : to save the file content and come out from the file

→ Esc + Shift + ZZ : to save the file content and come out from the file

→ Esc + shift : q ! : to come out from the file without saving the content

→ cc filename.c : to compile program

→ ./a.out : to run program

→ cc filename.c -o executable file name : to create separate executable file

→ ./executable file name : to run separate executable file.

→ rm filename.c : to remove file

→ Esc + yy : to copy lines in file

→ Esc + dd : to delete lines in file

→ cat : to create new file and add (append) data into old file, modification not possible.

→ cat > filename : to create new file

→ cat >> filename : to append data into old file

→ Ctrl + d : to come out from the file

→ cat filename : to view the file content.

→ cc -nostartfiles filename.c : to compile program without using main() function.

→ Preprocessor :-
cc -E filename.c -o filename.i
vi filename.i

→ Translator :-
cc -S filename.i -o filename.s

→ Assembler :-
cc -c filename.s -o filename.o
vi filename.o

→ Disassemble :- objdump : -D filename.o

→ Linker :-
cc filename.o
cc -S filename.c → Single translator upto
cc -c filename.c

→ man ASCII : ASCII table
→ man Operator : operators table

Date 19-6-20
Page

Session 4

→ Basic structure of C program
→ Compilation stages, errors.

```
#include <stdio.h>          // Header file
// global variables
// userdefined function prototype (declaration)
void main()
{
// local variables          // function call
// logic
}
// userdefined function definition
```

Two types of comments:-

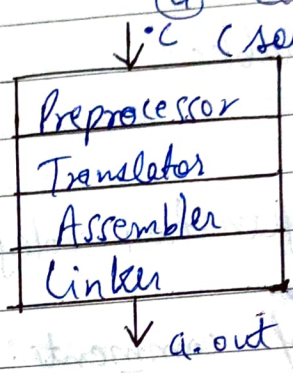
→ // single line comment
→ /* multi
line
comment */
→ wherever we want we can write the comment. Comment is not a part of program.

→ functions are the building blocks of a program. function is a set of instruction wraped/placed together to perform specific task.

→ main() is starting point of any program. ↑
is call by — start.

Compilation stages :-

- when we supply the program to the compiler, it will not convert this in one step, internally four stages takes place, so that it can be change into executable code or file your program will go in 4 stages
  ① Preprocessor ② Translator
  ③ Assembler ④ Linker



(1) Preprocessor :-

- Preprocessor is first stage in Compilation process. We will get pure C & extended file from source
- Preprocessor scans the source code & includes the header files which contain relevent information for various functions.
- preprocesor performs some tasks :-
  (i) Include header files
  (ii) Remove comments
  (iii) Replace macros
  (iv) Do conditional compilation
- Command to compile program upto preprocessor stage :
  cc -E filename
  cc -E filename.c -o filename.i

Why do we need to include header files?
→ C Programing language standard is before calling any fun. it should be prototyped or declared.

- To check the result of the preprocessor stage:
  vi filename.i

(ii) Translator:-
- Translator is second stage in compilation process.
- In this stage preprocessed code is translated to assembly instructions.
- This is responsible for checking the grammatical/syntax error. syntactical
- Translator tasks are:
  (i) Generate assembly code
  (ii) Checking for syntactical error
  It reports syntax error, if any
- Command to compile program upto Translator stage:
  cc -S filename.i -o filename.s
- To check the result of the preprocessor stage:
  vi filename.s

(iii) Assembler:-
- Assembler is third stage in compilation process.
- In this stage assembly program get converted into machine understandable code. ~~binary~~
- It is used to translate the assembly instructions to object code.
- Command to compile program upto Translator assembler Translator stage:
  cc -c filename.s -o filenam.o
- To check the result of the assembler stage:
  vi filename.o

Disassembler :- opcode to assembly

- The contents of this file (vi filename.o) are in binary formate (machine understandable)
- To make this file user understandable we need to disassemble it.
- Command to disassemble file:
  objdump -D filename.o

(iv) Linker:-
- Linker is fourthand last stage in compilation process.
- It this stage it generates executable code.
- Linker is responsible for linking with library.
- It will add OS information.

- Linker tasks are:-
  (i) Linking with library
  (ii) Searching function definition.
- Command to compile program upto translator stage:
  cc filename.o
- Single command to compile source code upto translator stage:
  cc -S filename.c
- Single command to compile source code upto assembly stage:
  cc -c filename.c

Session 5 :-

→ Errors in C program
→ Data types
  → char
  → int
  → Real (float, double)
  → string

Warning :- The compiler is giving a warning but it may generate error further. This shows because of expected output may not get by the compiler.

Errors :- While writing C programs, errors also known as bugs in the world of programming may occur unwillingly which may prevent the program to compile and run correctly as per the expectation of the programmer.
Basically two types of errors in C programming.
(i) Compile time error
(ii) Runtime error

(i) Compile time error :-

- Compile errors are those errors that occure at the time of completion of the program. C & it is generated by the compiler. C compiler time error may further classified as.
  ① Compile time preprocessor error
  ② Compile time translator error
  ③ Compile time linker error
  ④ Compile time assembler error.

① Preprocessor error:
- It is generated by preprocessor.
- It occures due to improper declaration of header file.
- Main this error is indicated as : fatel error on display i.e <STdio.h> cause error.

② Translator error:- It is generated by translator
- It occures due to synetactical mistakes or any syntax is missing.
- When the rules of C programming are not followed, the compiler will show syntax error.

③ Linker error:- It is generated by linker.
- It is occures due to improper declaration of functions / definition of functions.
- occures in both functions ; userdefined and predefined.

④ ~~It is generated by assembler~~

⑤ Assembler:- It is generated by assembler.
- It occures due to corrupted .s file (i.e translated assembly file)
- There is no chance of getting assembler error until & unless the .s file is purposely corrupted.

(i) ~~Compile time error~~ :-

(ii) Run time error :-
- Runtime errors are those errors that occur during the execution of a C program and generally occure due to some illegal operation performed in the program.
- Operating system will generate this error
- May be by dividing a no. by zero;
→ trying to open a file which is

not created, lack of free memory space.
It is divided into two types.
① Floating point exception error:
② Segmentation fault error.

① Floating point exception error:-
- When a program attempts to perform an illegal operation. such as dividing a no. by zero (a/0, 5/0) such error is called floating point error.

② Segmentation fault error:- It occures if I/P data given to the program is not in a correct format or I/P data is not found at specified path.
- If hardware problem occures such as hard disk error or hard disk full. or program tries to access other programs memory.

scanf("%d",a); error
proper address not provided

Section 6

→ Constants and Variables
→ Data types → 4
① → char
② → int
③ → Real (float, double)
④ → string

→ Whenever we are using scanf to scan the data from the keyboard we need to use proper format specifier (%d, %c etc) & proper address (&)

→ $ vi .vimrc
set nu Esc + shift zz (for row lines)