**DAY 06 (29-10-2024)**

* **ctags:** find out where the functions are declared
* sudo apt install universal-ctags   to install the ctags tool
* ctags -R . --- command used to generate the index data file i.e tags file
* set tags path of tag file --- in command mode in the vi file

* **navigating:**
* cltr[] --- to navigate (to search for the function declared in the current function)
* cltr+t --- to move backward

* **cscope**:
* sudo apt install cscope – to install the cscope tool
* rebuild the data base
* cscope.out file formed or produced
* cscope find . -name ‘\*.[ch]’
* cltr+d coming out of that csope

* **gprof**: to check amount of time consumed in different functions
* gcc -o application -pg filename
* gmon.out file is obtained after compiling above command
* executing gmon.out file we get g.profile file

* **gdb**: The GNU debugger - check the flow
* debugger used to check run time errors if any are present
* break – debugger utility
* watch – to keep an eye on the variable change or we can use the print to know how the variables are changing.
* gdb a.out command used for execution after compiling the file i.e gcc -g filename.
* Options used are break (b), run(r), continue(c), list (l), next (n), step (s) – to run the program inside the function

**C**

* Dennis Ritchie developed c language in 1972. It took him 3 years 8 months to develop this language.
* Business package- notepad, excel, IRCTC

System software- device drivers, software commands like date, time

C is suitable for both

* C is **faster and efficient** because other languages need virtual box unlike C and file size is lesser
* C is suited for **structured programming**
* Highly portable

* **Structure of c program**

1. Documentation section: user purpose

/\*

Description: program details, algorithms, libraries used are mentioned

Author: CSP

Date of creation (DOC)/ Date of modification (DOM):

Version: 0.1v- new creation

             Usually, 1.0 is used as stable version with no bugs and errors

\*/

1. Link section: including header files

* Libraries references are linked in link session

1. Definition: prototype of function, macros definition



1. Global declaration section: variables are written and accessed

* Variable and function names should be meaningful to maintain coding standards
* Ideally function names camel cases

1. Main () function section:
2. Declaration part: all variables should be declared at first
3. Executable part: executing expression, function (anything performing task)

* Every function has its own address same goes with main function also
* We can have more than one main function
* commands
* Carriage return \r – cursor returns to the start of the same line we are working

New line \n - cursor points to the start of the next line

\b – backspace the values

* int x=10, y=20;

printf(“%d”, x, y); ---- o/p: 10

* C tokens: keywords, identifiers, constants, strings, special symbols, operators
* Naming convention shouldn’t use numeric in the start of the variable
* anything in “ “ is  string, ‘ ‘ is a single character
* mod operator can’t be operated on float/double
* operators
* datatypes – fundamental, derived (array, function, strings), user defined
* modifiers – signed, unsigned, long, short
* identifiers – identify the variables, functions and other user defined variables
* the variables cannot be start with numeric, $ but can be started  \_, letters.
* int a=b=c=10;

a=b=c=50;

leads to the compile error we can’t assign values like this is C