**STATIC ANALYSIS USING SPLINT:**

int \*ptr;

printtf(“%d”, \*ptr);

* A simple code review shows we are accessing an invalid ptr (which has not been malloced or pointed to a valid memory area).
* Static analysis tools, analyze the code in similar (but much more sophisticated) ways, and warn us of potential bugs in code.
* They are called “static” analyzers, as they analyze the stationary (not running) code.

**Installing Splint:** $sudo apt install splint

EX:

#include<stdio.h>

int main()

{ int \*ptr; (or) int \*ptr=NULL;

printf(“%d”, \*ptr);

return 0;

}

**Splint t2.c** ----variable ptr used before definition.

**EX:**

#include<stdio.h>

#include<stdlib.h>

int main()

{ int \*ptr=(int\*)malloc(sizeof(int));

\*ptr=10;

printf(“%d”, ptr);

free(ptr);

return 0; }

gcc -wall t2.c ---- compile

./a.out

* Bug-------Segmentation fault(core dumped)

|

* Allocate the value without the address(memory allocate).

**C Tages:** How &where the function os declared.

* Installing C TAGS---- $sudo apt install universal-ctags
* Generating index file: $ctags-R.
* Loading the index data in vi: $vi file1.c#relevant.file name
* In vi run below command: set tags=<path of -tag-file>:set tags=../..tags
* Navigating : press control + ]
* Backwards: presss control + t
* Stack : Last In First Out

**ESCOPE:**

* Install---$sudo apt install escope
* Generating the escope database – escope.out
* It rebuilds the database incrementally.
* $escope ‘find, -name.”[ch]”
* (or)
* $find .-name”.[ch]” >escope.files
* Coming out---- control+D
* Escope -I escope.files
* ps----to see all

**PERFORMANCE PROFILING USING gprof:**

* gprof allows us to measure how much percentage of time is spent in different functions.This way, we can focus on optimizing the functions which take more time or which are called more number.
* $gcc -o application -pg main.c
* Run----$./application-----gmon.out contains the performance data.
* $grof ./application gmon.out

**GDB**-------The GNU Debugger

* Debugging : To know the flow of a program .----Break Point

-----Watch

-----Navigation (Next or n)

-----step(s)

* List----list the code of lines in source code while running.
* quit----exit from gdb.
* Gcc -g main.c------compile
* gdb ./a.out (or) gdb a.out ----run
* b main
* b function1