Frama-C Installation Tutorial

(FRAmework for Modular Analysis of C code)

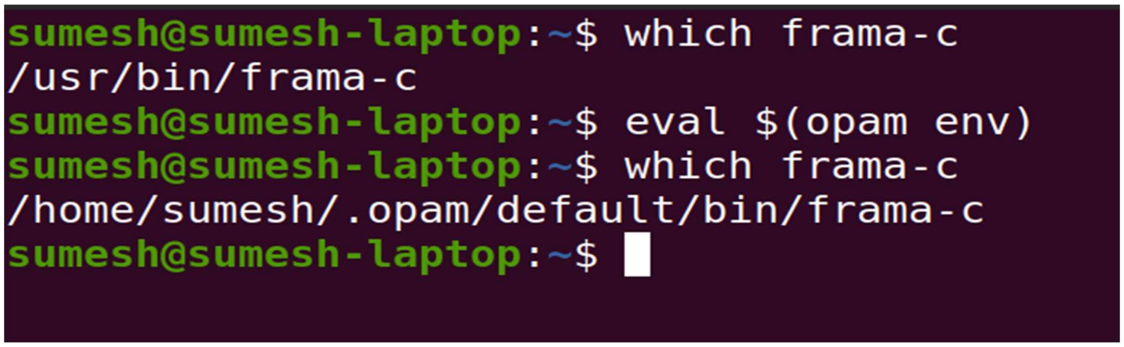
Installing Frama-C via Opam (OCaml library PAckage Manager): -

Open **Terminal** in Ubuntu and Execute the following commands one after the other.

1. **$ sudo apt install opam**	; Install Opam
2. **$ opam init**	; To initialize Opam
3. **$ opam install depext**	; Install depext tool of Opam for dependencies
4. **$ eval $(opam env)**	; To update the current shell environment
5. **$ opam depext frama-c**	; Install dependencies of Frama-c using depext tool
6. **$ opam install frama-c**	; Install Frama-c using Opam
7. **$ why3 config detect**	; Configure Why3 to detect automated provers

Testing Frama-C installation: -

Using command: $ **which frama-c**



```
sumesh@sumesh-laptop:~$ which frama-c
/usr/bin/frama-c
sumesh@sumesh-laptop:~$ eval $(opam env)
sumesh@sumesh-laptop:~$ which frama-c
/home/sumesh/.opam/default/bin/frama-c
sumesh@sumesh-laptop:~$ 
```

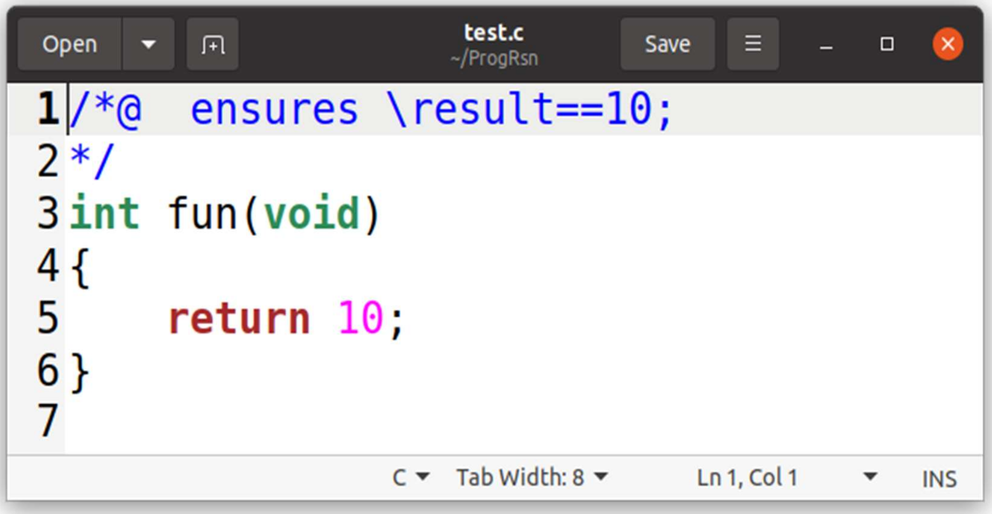
References: -

https://frama-c.com/html/get-frama-c.html#

https://git.frama-c.com/pub/frama-c/blob/master/INSTALL.md#installing-frama-c

Working with Frama-c: -

1. Type the following code using **gedit** in Ubuntu and save it as **test.c**



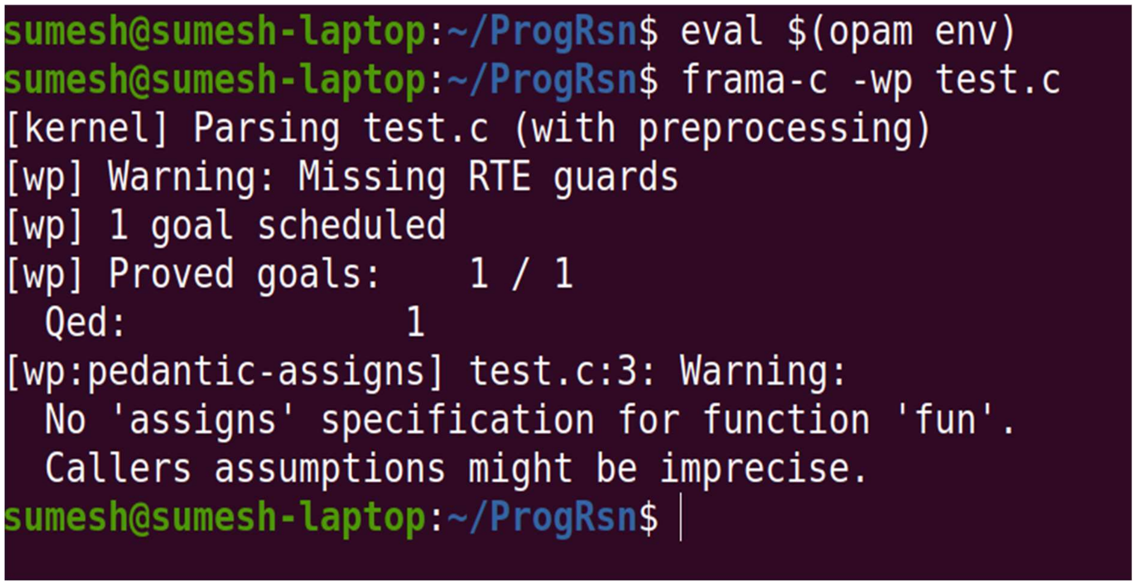
```c
/*@  ensures \result==10;
*/
int fun(void)
{
    return 10;
}

```

2. Open the **Terminal** and set **opam env**ironment by running following command.

   $ eval $(opam env)

3. Now run test.c using **frama-c** tool with **wp plugin** using the command.

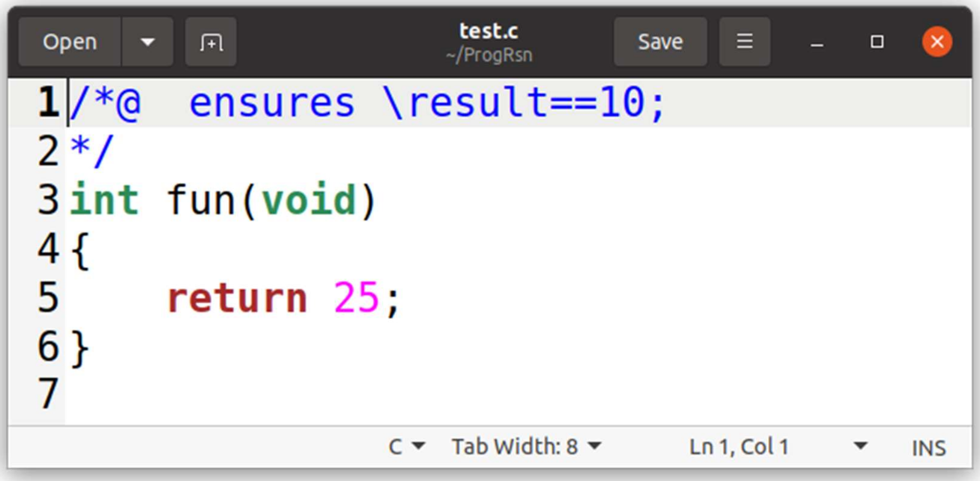
   $ frama-c -wp test.c



```
sumesh@sumesh-laptop:~/ProgRsn$ eval $(opam env)
sumesh@sumesh-laptop:~/ProgRsn$ frama-c -wp test.c
[kernel] Parsing test.c (with preprocessing)
[wp] Warning: Missing RTE guards
[wp] 1 goal scheduled
[wp] Proved goals:    1 / 1
  Qed:             1
[wp:pedantic-assigns] test.c:3: Warning:
  No 'assigns' specification for function 'fun'.
  Callers assumptions might be imprecise.
sumesh@sumesh-laptop:~/ProgRsn$ 
```

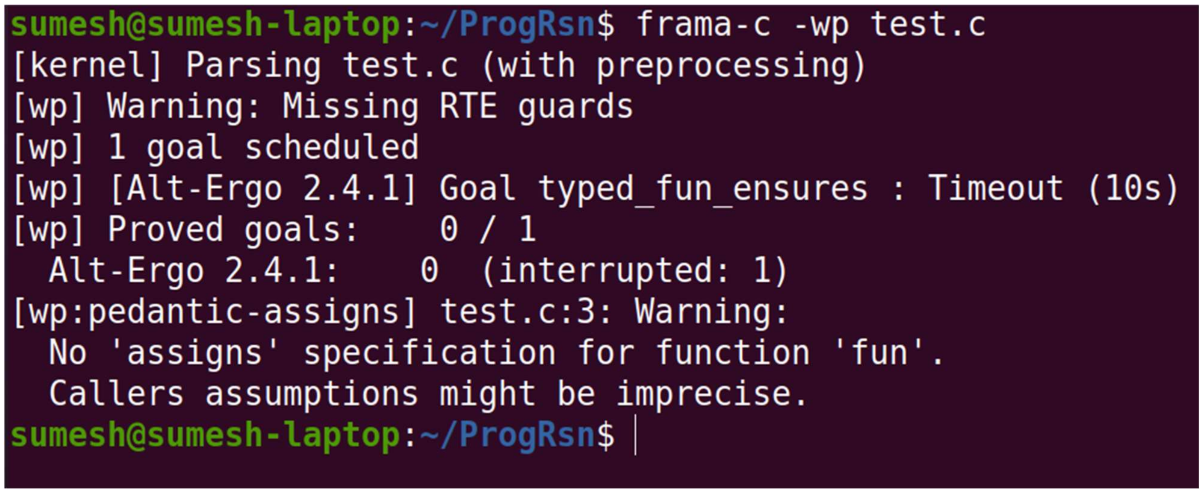
Proved 1/1 goals indicate functional objectives achieved.

4. Now change the code test.c as given below.



```c
/*@  ensures \result==10;
*/
int fun(void)
{
	return 25;
}

```

5. Run again using **frama-c.**



```
sumesh@sumesh-laptop:~/ProgRsn$ frama-c -wp test.c
[kernel] Parsing test.c (with preprocessing)
[wp] Warning: Missing RTE guards
[wp] 1 goal scheduled
[wp] [Alt-Ergo 2.4.1] Goal typed_fun_ensures : Timeout (10s)
[wp] Proved goals:    0 / 1
  Alt-Ergo 2.4.1:    0  (interrupted: 1)
[wp:pedantic-assigns] test.c:3: Warning:
  No 'assigns' specification for function 'fun'.
  Callers assumptions might be imprecise.
sumesh@sumesh-laptop:~/ProgRsn$
```

Proved 0/1 goals indicate functional objectives NOT achieved.

Working with Frama-c GUI: -

Run test.c using **frama-c-gui** tool and **analyze** output before and after modification.

$ frama-c-gui -wp test.c

****