Why inline function needs static storage class ?

Inline Function are those function whose definitions are small and be substituted at the place where its function call is happened.

Function substitution is totally compiler choice.

#include <stdio.h>

// Inline function in C

inline int foo()

{

return 2;

}

// Driver code

int main()

{

int ret;

// inline function call

ret = foo();

printf("Output is: %d\n", ret);

return 0;

}

Compiler Error:

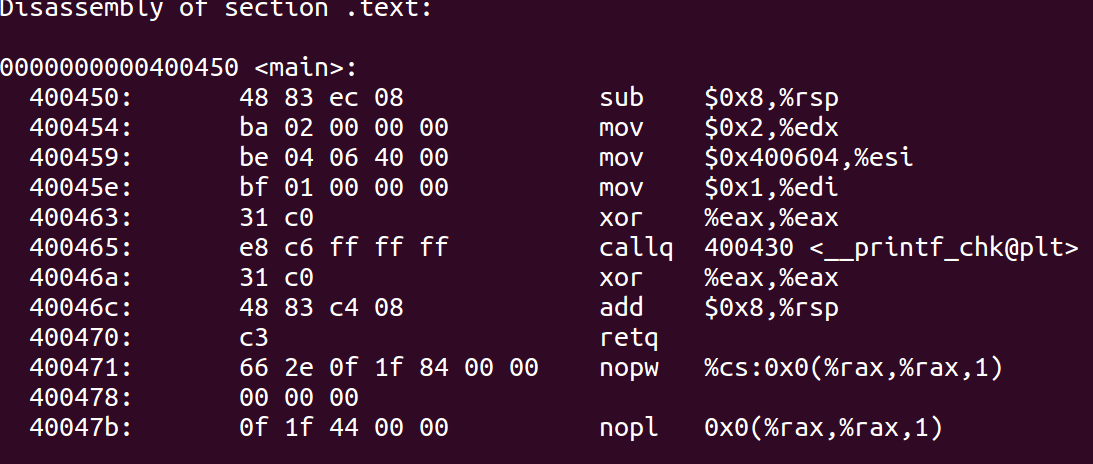
In function `main':

undefined reference to `foo'

Why this error happened?

This is one of the side effect of GCC the way it handle inline function. When compiled, GCC performs inline substitution as the part of optimisation.

So there is no function call present (foo) inside main. Please check below assembly code which compiler will generate.



Normally GCC’s file scope is “not extern linkage”. That means inline function is never ever provided to the linker which is causing linker error, mentioned above.

**How to remove this error?**

To resolve this problem use “static” before inline. Using static keyword forces the compiler to consider this inline function in the linker, and hence the program compiles and run successfully.

#include <stdio.h>

// Inline function in C

static inline int foo()

{

return 2;

}

// Driver code

int main()

{

int ret;

// inline function call

ret = foo();

printf("Output is: %d\n", ret);

return 0;

}

Output is: 2