Libft

Why include header file in source code?

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/836417251913236533/836417252302782505/863358784079265802

Tips for Libft In General

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/836417251913236533/836417252302782506/863649525858959379

Seeing the Makefile in Action

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/836417251913236533/836417252302782505/865252439846027294

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/836417251913236533/866165728382156820/866166185515417620

Imperative vs Declarative Programming - ui.dev

You've undoubtedly heard about imperative programming vs. declarative programming. You might have even searched for what those terms actually mean. Sadly, you probably encountered a definition similar to this "You know, imperative programming is like how you do something, and declarative programming is more like what you

https://ui.dev/imperative-vs-declarative-programming/

Function prototypes

https://www.ibm.com/docs/en/rbd/9.5.1?topic=functions-function-prototypes

https://www.ibm.com/docs/en/rbd/9.5.1?topic=functions-function-prototypes

Testers for Libft

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/863788373604827168/863788374082715700/869550624033886288

When to use void

Discord - A New Way to Chat with Friends & Communities

Discord is the easiest way to communicate over voice, video, and text. Chat, hang out, and stay close with your friends and communities.

https://discord.com/channels/836417251913236533/836417252302782506/869266544352456774

Notes On Libft

Project Requirements

Program name	libft.a
Turn in files	*.c, libft.h, Makefile
Makefile	Yes
External functs.	Detailed below
Libft authorized	Non-applicable
Description	Write your own library, containing an extract of important functions for your cursus.

Red text - yet to be answered

Notes On Header Files and Compilation

1. What is the use of .a file?
    a. .a file refers to static library that contains all the standard C functions that are commonly used. It's created with the Linux tool, ar. Why and when to use static vs dynamic library?

A File Extension - What is an .a file and how do I open it?

An A file contains a library of functions and headers that may be referenced by a C/C++ source file. It may store only a few functions or may include an entire library of functions, such as a 3D modeling engine. A files are typically created by the GNU ar utility.

https://fileinfo.com/extension/a

Static and Dynamic Libraries | Set 1 - GeeksforGeeks

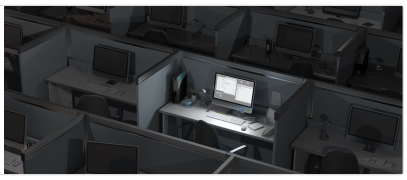
When a C program is compiled, the compiler generates object code. After generating the object code, the compiler also invokes linker. One of the main tasks for linker is to make code of library functions (eg printf(), scanf(), sqrt(), ..etc) available to your program.

https://www.geeksforgeeks.org/static-vs-dynamic-libraries/

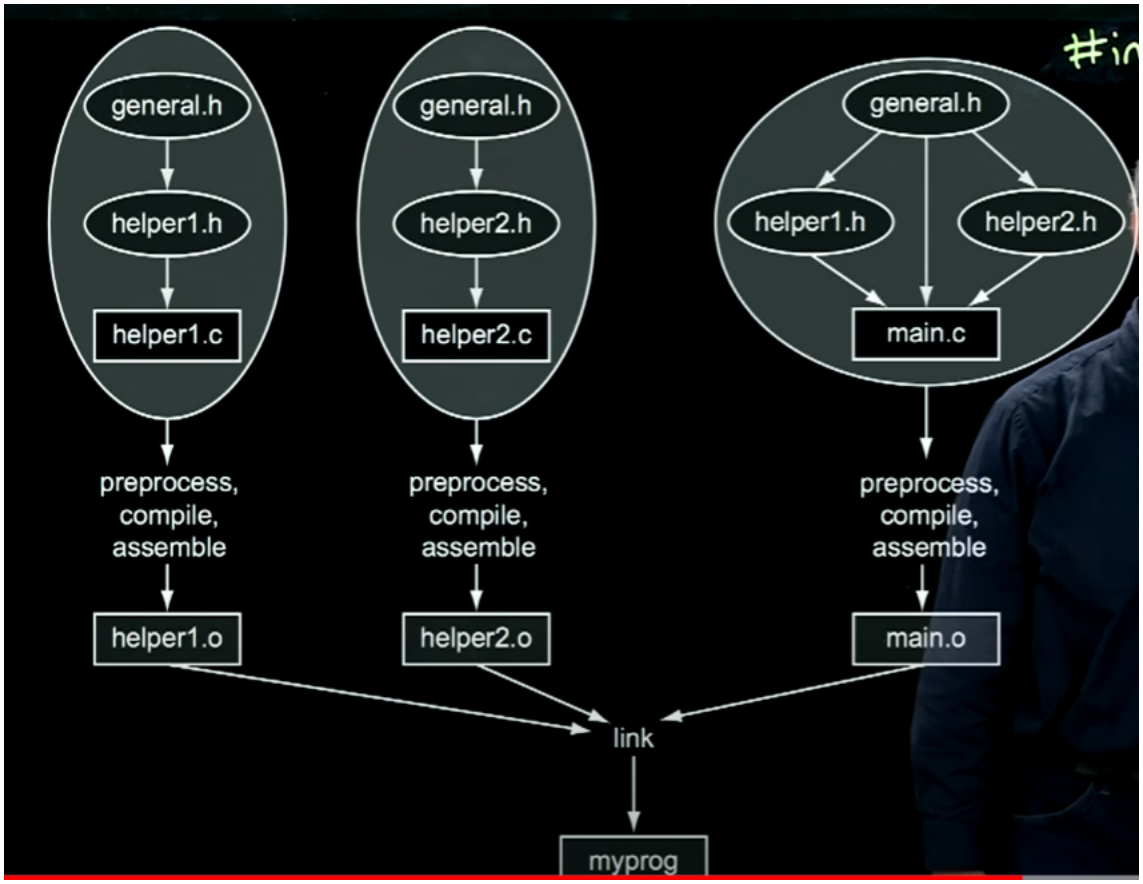
What You Need to Know About the Linux/Unix Command: Ar

The GNU ar program creates, modifies, and extracts items from file archives. An is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

https://www.lifewire.com/ar-linux-command-4093866



2. How does compilation in C work?

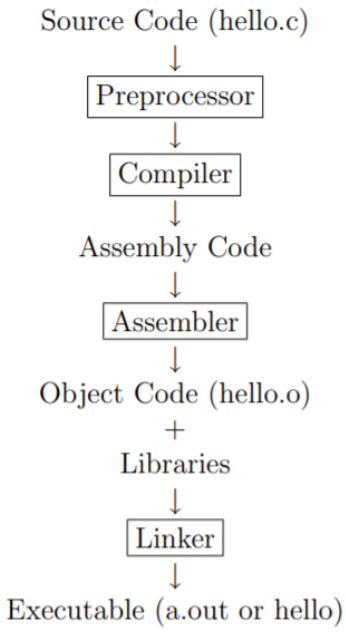


1. At the end of Executable, there is also a Loader

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/edda121f-5468-4f0f-bb1a-b4734d16372e/c_program_compilation.pdf

The C compilation model

When you invoked the `gcc` compiler, a series of steps were performed in order to generate an executable from the text you entered as your C program. These steps are as follows



https://www.geeksforgeeks.org/linker/

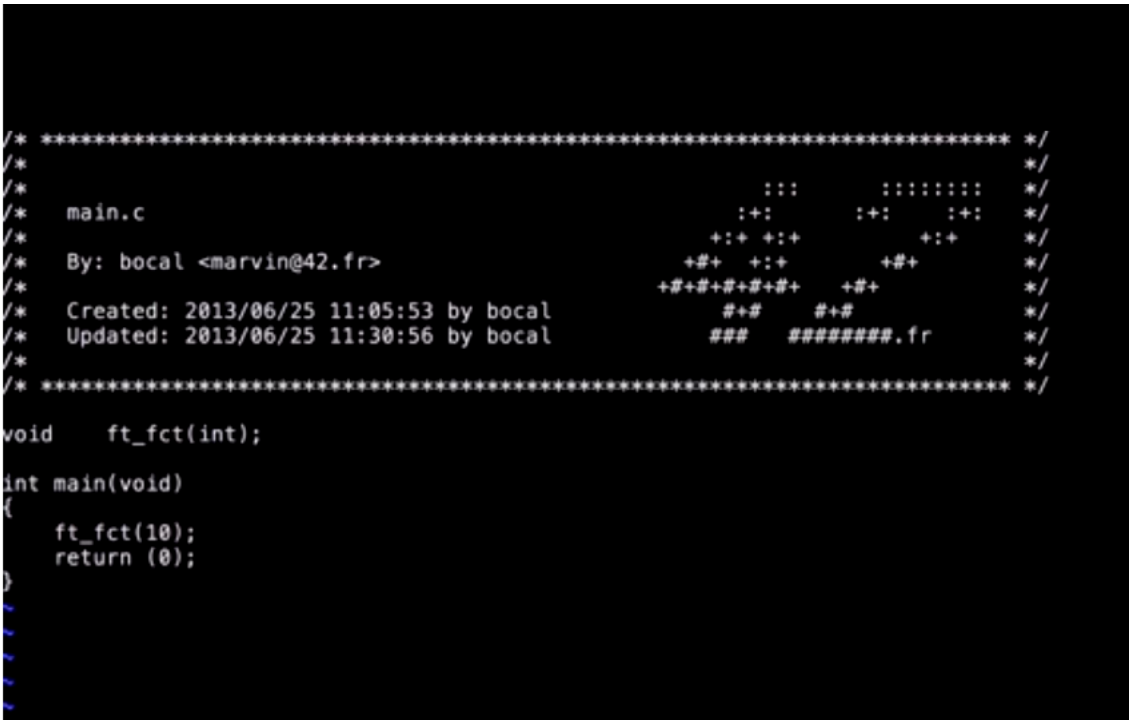
3. Why linking is still needed if header files are already included during preprocessing?
4. How does linking actually work to produce an executable?
5. Which operation occur during run time vs compile time?
6. What is a header file? https://www.learncpp.com/cpp-tutorial/header-files/#:~:text=The primary purpose of a,propagate declarations to code files.&text=Header files allow us to,typing in multi-file programs.
    a. Contains macros, it's a piece of code with a predefined keyword or function that the programmer can write to customize specific behavior. When the keyword or function is used in source file, it will replace the variables in the source code with the behavior of the macros. It also contains only declarations of functions, not the implementation of it.
    b. https://gcc.gnu.org/onlinedocs/cpp/Header-Files.html#:~:text=A header file is a,shared between several source files.&text=Your own header files contain,source files of your program.
    c. Header files contain only function declarations because multiple source files can include the same header. Hence, if definition of functions are included, linker will throw an error since multiple definitions are found. In my POV, multiple definitions of the same function isn't good because it makes debugging hard.
7. Why use header files?
    a. For ease of maintaining the functions that are used by multiple files. If the same function is copied in multiple files, I would need to change each the function that appears in each of the files. But with using headers, I can skip this step. Plus this makes compilation and linking time quicker since we aren't unnecessarily including previously included header.

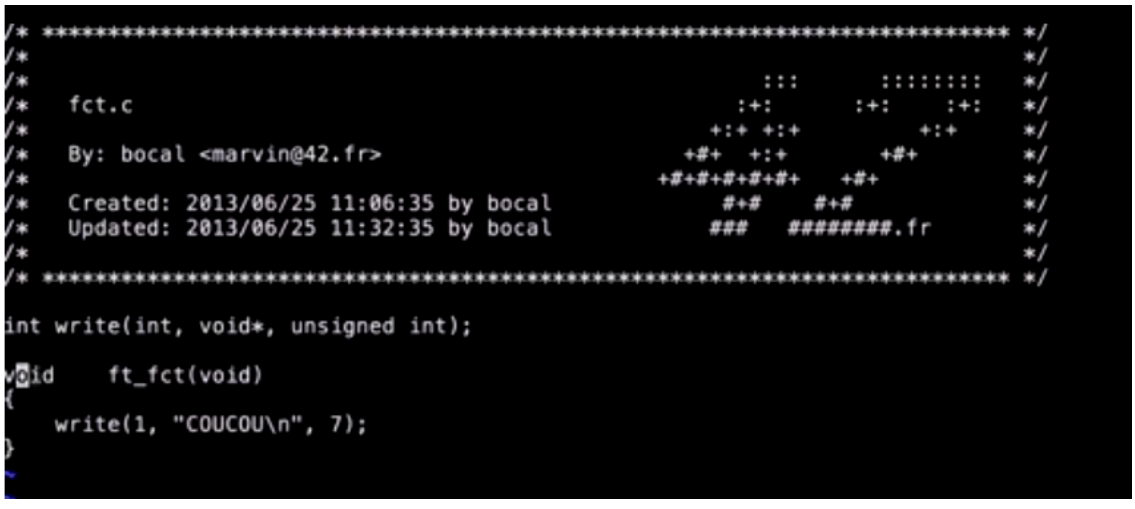
b. Acts as a point of reference for source file containing the function and the file calling the function. This will ensure that when the function prototype in either the header or the source file is changed, error can be detected and corrected.

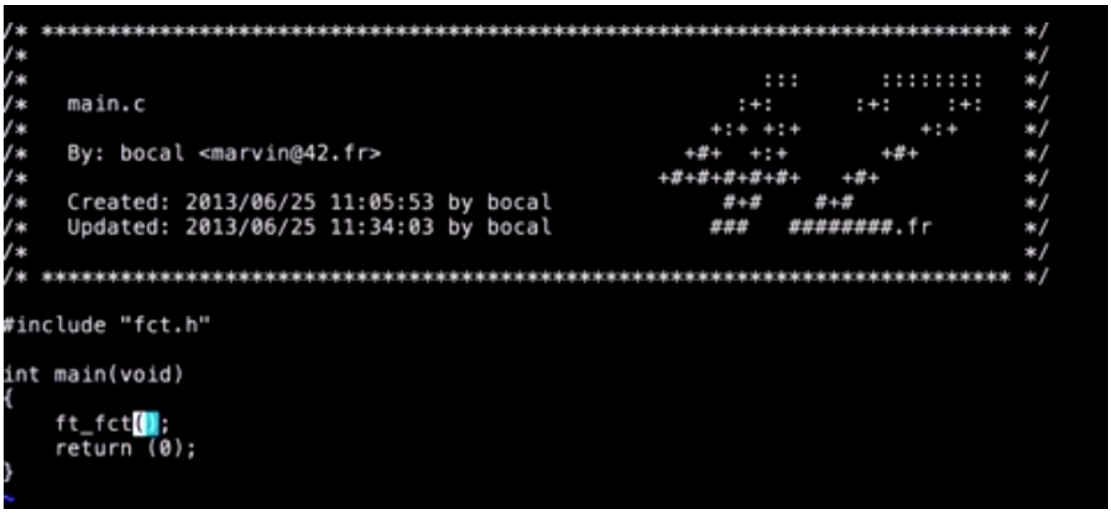
c. For security or resource conservation purposes? Why would a linker throw an error when it sees the same function being defined twice (declaring a function means writing its prototype. To define means to write the instructions for the function)?

   i. From the POV of security. I can be using a function that has its prototype defined differently and still run the file. This can be solved by using a header file that will throw a compilation error when it detects that the function is being referenced correctly.

   ii. Next question, why only when a header file is used the compiler will throw an error on different data types declared for the same function name?







d. How does using header guards solve the linking error when two of the same functions have been defined? https://www.learncpp.com/cpp-tutorial/header-guards/

   i. This can be solve through conditional inclusion written in macros, where if the header that contains the function has already been defined, it wouldn't define it again. This is useful for preventing more than one definition of function that can occur when there is interdependency among header files. Example, car.h includes engine.h. Subsequently, main.c includes both car.h and engine.h. Without header guards, main.c will have 2 copies of engine definition.

   ii. How does conditional inclusion of header and having only function declarations in header files allow multiple C source code files to share the same header (i.e #include in multiple C source files)?

      i. Header guards only prevent the same header from being included more than once in the same source file. It doesn't prevent multiple source files from including it.

   iii. However, usually header only contains declarations, hence, even if it's included more than once, it wouldn't be a problem. Example below, ded.h includes test.h. Hence, the hello.c will have two test.h included but the program still runs.

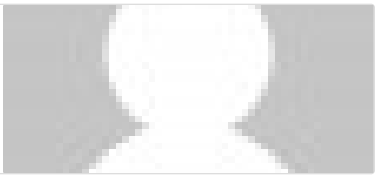
   iv. Under what conditions will header file include definitions?

https://stackoverflow.com/questions/3805787/defining-a-function-twice-in-c/3805833

2.11 - Header guards

The duplicate definition problem In lesson 2.6 -- Forward declarations and definitions, we noted that a variable or function identifier can only have one definition (the one definition rule).

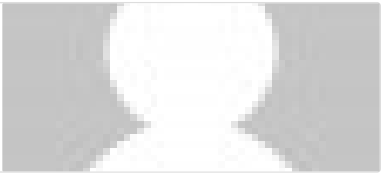
https://www.learncpp.com/cpp-tutorial/header-guards/



2.11 - Header guards

The duplicate definition problem In lesson 2.6 -- Forward declarations and definitions, we noted that a variable or function identifier can only have one definition (the one definition rule).

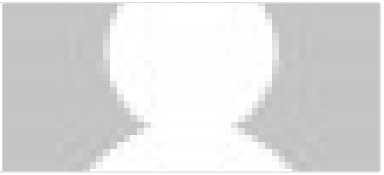
https://www.learncpp.com/cpp-tutorial/header-guards/



2.11 - Header guards

The duplicate definition problem In lesson 2.6 -- Forward declarations and definitions, we noted that a variable or function identifier can only have one definition (the one definition rule).

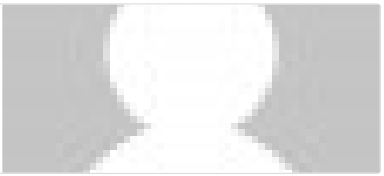
https://www.learncpp.com/cpp-tutorial/header-guards/



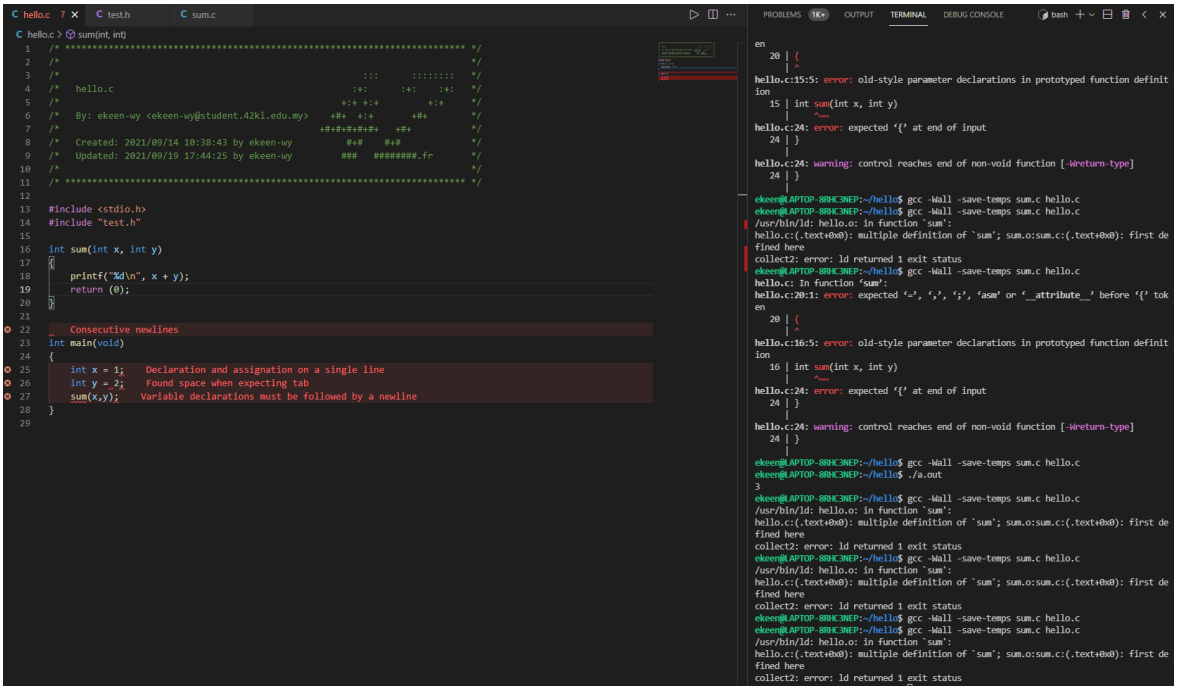
2.11 - Header guards

The duplicate definition problem In lesson 2.6 -- Forward declarations and definitions, we noted that a variable or function identifier can only have one definition (the one definition rule).

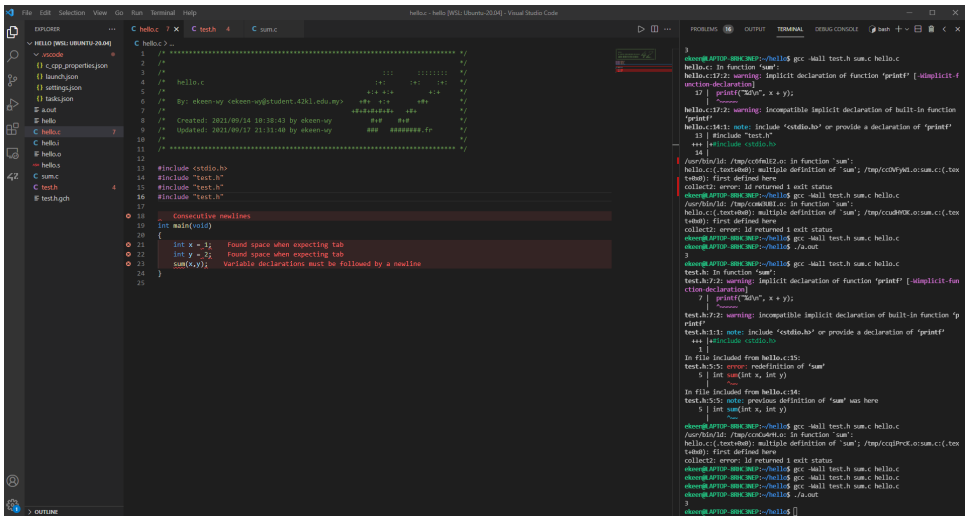
https://www.learncpp.com/cpp-tutorial/header-guards/

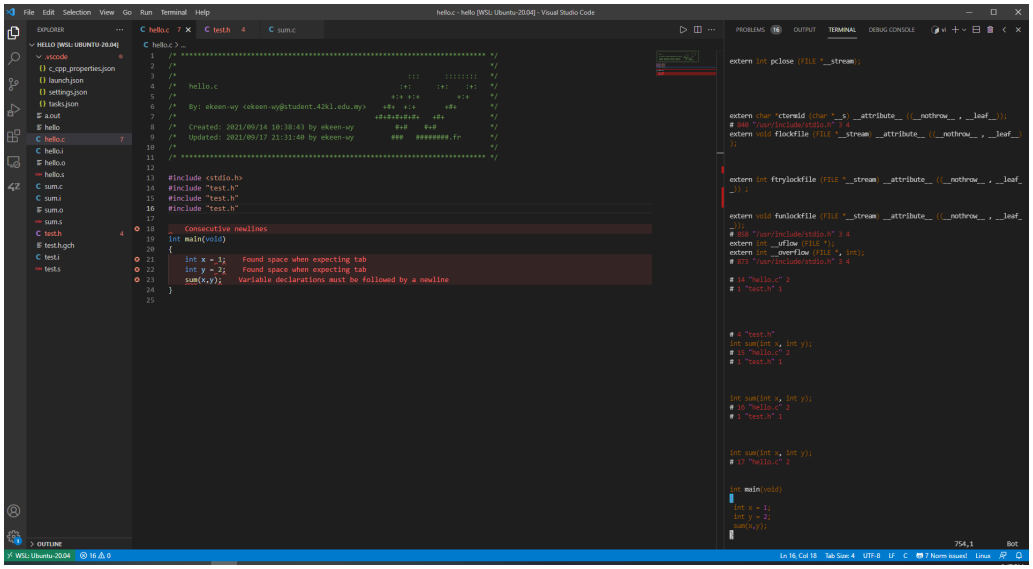


8. The picture below has a function definition. My include test.h file only has a declaration. How come the compiler will still throw a multiple definition error? How does the header file pull the codes from source file?
    a. This error occurs because in my hello.c I have sum function defined and in sum.c I have it defined also. When compiling both these files in together in the same compilation unit, an error occurs.



9. Why I don't need to compile header files when compiling C files?
    - Because the preprocessor will have included the prototypes before compilation.
10. Does a library contain header files or only object files?





> Headers are not stored in libraries. Headers are stored separately from libraries. Libraries contain object files; headers are not object files. By default, standard headers on a Unix system are stored in /usr/include — you'll normally find /usr/include/stdio.h and /usr/include/string.h and /usr/include/stdlib.h, for example. By default, libraries are stored in /usr/lib (but you may also find some in /lib). Often, compilers are configured to look in some other places too. One common alternative location is under /usr/local, so /usr/local/include for headers and /usr/local/lib for libraries. Note, too, that a single library may have many headers defining the services. The default library is an example. It has the functions corresponding to those found in <stdio.h>, <string.h>, <stdlib.h> and many other headers too.

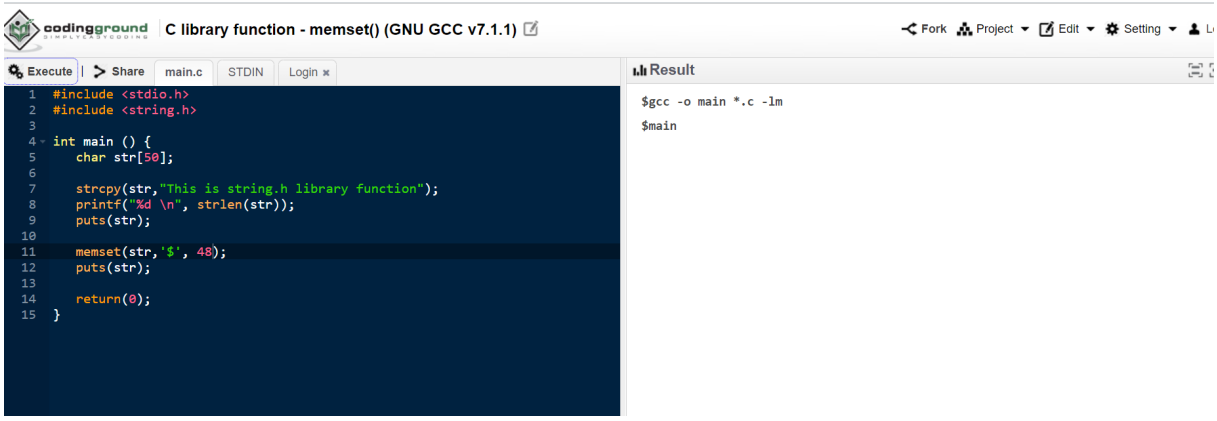
Including header file from static library

I am making a test setup of a C static library and program.

https://stackoverflow.com/questions/27660713/including-header-file-from-static-library#:~:text=Headers%20are%20not%20stored%20in,%2Fusr%2Finclude%2Fstdio

Miscellaneous

1. Why does setting the memset to 48 doesn't print anything (it went out of bound)?



2. Why does the return value can be negative when I casted it to unsigned char? Why do I need to cast this to unsigned for strncmp's return value?

```c
#include "libft.h"

int ft_strncmp(char *s1, char *s2, unsigned int n)
{
    unsigned int    i;

    i = 0;
    while ((s1[i] != '\0' || s2[i] != '\0') && i < n)
    {
        if (s1[i] != s2[i])
        {
            return ((unsigned char) s1[i] - s2[i]);
        }
        i++;
    }
    return (0);
}
```

main.c

```
20
21  int main(void) {
22    char arr1[] = "tests";
23    char arr2[] = "testss";
24    //int i = 0;
25
26    //while (arr2[i] != '\0')
27    //{
28      //printf("%c\n%d\n", arr1[i], i);
29      //i++;
30    //}
31    printf("%d\n", ft_strncmp(arr1, arr2, 6));
32  }
33
34
```

Console Shell

```
clang-7 -pthread -lm -o main main.c
./main
-115
```

3. The error below means that because int's max positive number can never match long's max value, the conditional can never be true.

ft_atoi.c

Libft > ft_atoi.c > ft_atoi(char *)

```
13  #include "libft.h"
14
15  int ft_atoi(char *str)
16  {
17      int i;
18      int sign;
19      int base_conv;
20
21      i = 0;
22      sign = 1;
23      base_conv = 0;
24      if ((str[i] != '+' || str[i] != '-') && (str[i]
25          return (0);
26      if (str[i] == '-')
27      {
28          sign *= -1;
29          i++;
30      }
31      if (str[i] == '+')
32          i++;
33      while (str[i] != '\0' && (str[i] >= 0 && str[i]
34      {
35          base_conv = base_conv * 10 + (str[i] - '0');
36          i++;
37      }
38      if ((long) base_conv > 9223372036854775806)
39          return (-1);
40      return (base_conv * sign);
41  }
42
```

PROBLEMS 631 OUTPUT TERMINAL ... bash - libft-unit-test

```
 make -j 3 -C ../Libft
make[1]: Entering directory '/home/ekeen/42KL_Core/Libft'
cc -Wall -Wextra -Werror   -c -o ft_atoi.o ft_atoi.c
ft_atoi.c: In function ‘ft_atoi’:
ft_atoi.c:33:19: error: multi-character character constant [-We
rror=multichar]
   33 |  while (str[i] != '/0' && (str[i] >= 0 && str[i] <= '9'
))
      |                   ^~~~
ft_atoi.c:33:16: error: comparison is always true due to limite
d range of data type [-Werror=type-limits]
   33 |  while (str[i] != '/0' && (str[i] >= 0 && str[i] <= '9'
))
      |                ^~
ft_atoi.c:38:16: error: comparison is always false due to limit
ed range of data type [-Werror=type-limits]
   38 |  if (base_conv > 9223372036854775806)
      |                ^
cc1: all warnings being treated as errors
make[1]: *** [<builtin>: ft_atoi.o] Error 1
make[1]: Leaving directory '/home/ekeen/42KL_Core/Libft'
make: *** [Makefile:179: assets/libft.a] Error 2
ekeen@LAPTOP-8RHC3NEP:~/42KL_Core/libft-unit-test$ make f
 make -j 3 -C ../Libft
make[1]: Entering directory '/home/ekeen/42KL_Core/Libft'
cc -Wall -Wextra -Werror   -c -o ft_atoi.o ft_atoi.c
ft_atoi.c: In function ‘ft_atoi’:
ft_atoi.c:38:23: error: comparison is always false due to limit
ed range of data type [-Werror=type-limits]
   38 |  if ((long) base_conv > 9223372036854775806)
      |                       ^
cc1: all warnings being treated as errors
make[1]: *** [<builtin>: ft_atoi.o] Error 1
make[1]: Leaving directory '/home/ekeen/42KL_Core/Libft'
make: *** [Makefile:179: assets/libft.a] Error 2
ekeen@LAPTOP-8RHC3NEP:~/42KL_Core/libft-unit-test$
```

4. How does the void parameter actually work? Because I am passing in a char arr into the void *s parameter, and I now have to declare char pointer to hold the address of a char? How does the backend work? And what is the difference between using an unsigned char vs char pointer to hold the address?

```c
#include "libft.h"

void    *ft_memset(void *s, int c, size_t n)
{
    char    *ptr;

    ptr = s;
    while (n-- != 0)
    {
        *ptr = c;
        ptr++;
    }
    return (s);
}
```

https://www.journaldev.com/36863/memset-in-c#:~:text=Syntax of memset() in,is again a void* pointer

5. The code below wouldn't return the location of dest accurately because when incrementing the pointer, I have moved the location where it's pointing to. Instead, assign dest's address to another pointer and increment that pointer.

```c
#include "libft.h"

void    *ft_memcpy(void *dest, const void *src, size_t n)
{
    while (n-- != 0)
    {
        *(unsigned char*) dest = *(unsigned char*) src;
        dest++;
        src++;
    }
    return (dest);
}
```

```c
void      test_ft_memcpy_return(void *ptr) {
  typeof(memcpy)  *ft_memcpy = ptr;
  SET_EXPLANATION("your memcpy's return is false/doesn't work with basic params");

  SANDBOX_RAISE(
      char  src[] = "test basic du memcpy !";
      char  buff1[22];

      char  *r1 = memcpy(buff1, src, 22);
      char  *r2 = ft_memcpy(buff1, src, 22);
      SET_DIFF(r1, r2);
      ASSERT_RETURN_VALUE(r1, r2);

      r1 = memcpy("", src, 0);
      r2 = ft_memcpy("", src, 0);
      ASSERT_RETURN_VALUE(r1, r2);
      exit(TEST_SUCCESS);
      );

}
```

6. What is NULL?

NULL pointer in C - GeeksforGeeks

At the very high level, we can think of NULL as a null pointer which is used in C for various purposes. Some of the most common use cases for NULL area) To initialize a pointer variable when that pointer variable isn't assigned any valid memory address yet.b) To check for a null pointer before accessing any pointer variable.

https://www.geeksforgeeks.org/few-bytes-on-null-pointer-in-c/

https://wiki.sei.cmu.edu/confluence/display/c/EXP34-C.+Do+not+dereference+null+pointers#:~:text=Compliant Solution&text=Passing a null pointer to,other than checking for null

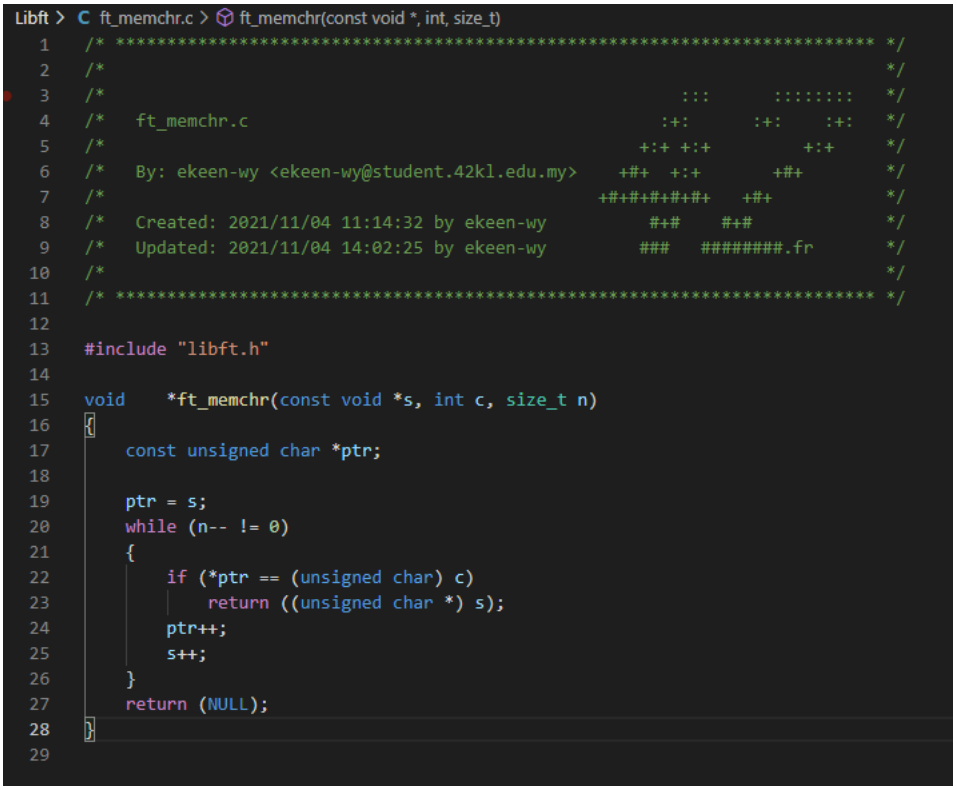
ImperialViolet

The C standard (ISO/IEC 9899:2011) has a sane-seeming definition of memcpy (section 7.24.2.1): The memcpy function copies n characters from the object pointed to by s2 into the object pointed to by s1. Apart from a prohibition on passing overlapping objects, I think every C programmer understands that.

https://www.imperialviolet.org/2016/06/26/nonnull.html

https://www.thoughtco.com/definition-of-null-958118#:~:text=In computer programming%2C null is,pattern for a null pointer.

7. If the return type is a pointer to void (a memory address), why would casting it to a unsigned char * not cause and error. Because I had to do this casting since the parameter s is a const void *.

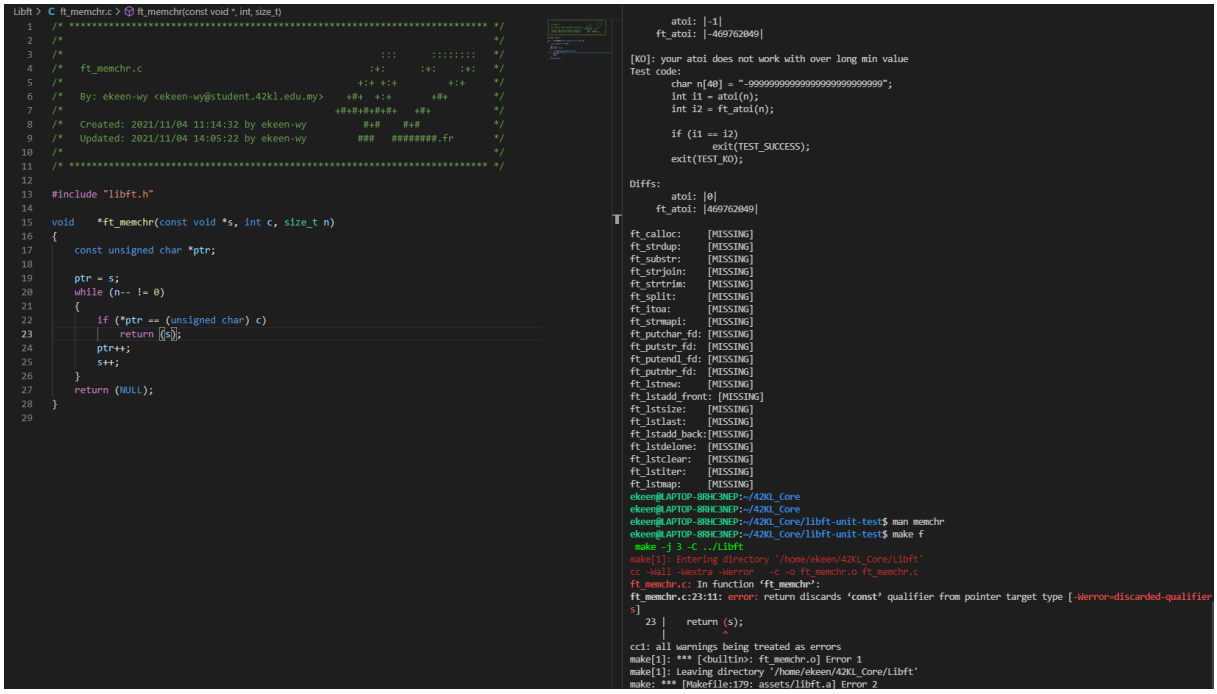


8. What does “return discards qualifiers from pointer” error mean?

C: Compiler warning "return discards qualifiers from pointer target type"

Thanks for contributing an answer to Stack Overflow! Please be sure to answer the question. Provide details and share your research! Asking for help, clarification, or responding to other answers. Making statements based on opinion; back them up with references or personal experience. To learn more, see our tips on

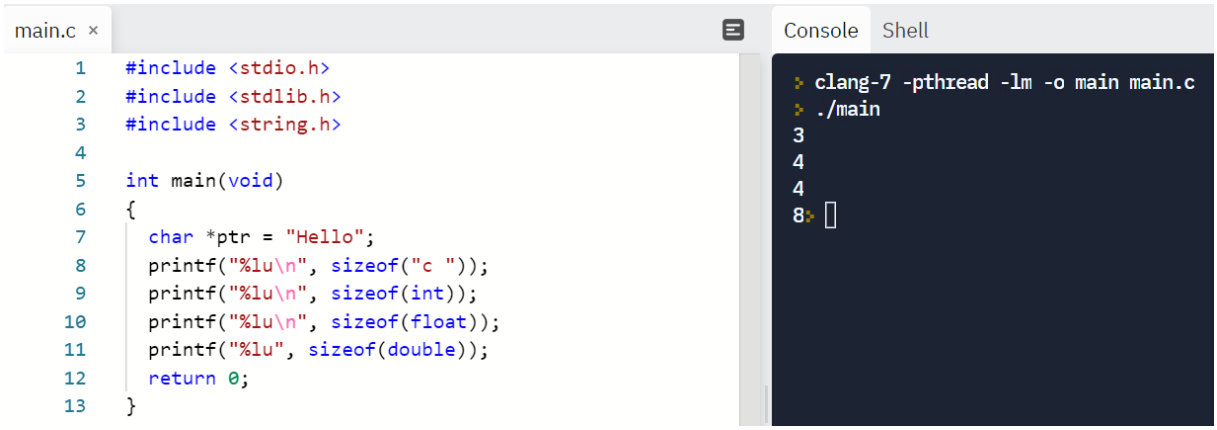
https://stackoverflow.com/questions/24830335/c-compiler-warning-return-discards-qualifiers-from-pointer-target-type/24830370#24830370



```
Libft > C ft_memchr.c > ft_memchr(const void *, int, size_t)
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_memchr.c                                        :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: ekeen-wy <ekeen-wy@student.42kl.edu.my>    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2021/11/04 11:14:32 by ekeen-wy          #+#    #+#             */
/*   Updated: 2021/11/04 14:05:22 by ekeen-wy         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include "libft.h"

void    *ft_memchr(const void *s, int c, size_t n)
{
    const unsigned char *ptr;

    ptr = s;
    while (n-- != 0)
    {
        if (*ptr == (unsigned char) c)
            return ((s));
        ptr++;
        s++;
    }
    return (NULL);
}
```

```
       atoi: |-1|
    ft_atoi: |-469762049|

[KO]: your atoi does not work with over long min value
Test code:
       char n[40] = "-99999999999999999999999999";
       int i1 = atoi(n);
       int i2 = ft_atoi(n);

       if (i1 == i2)
           exit(TEST_SUCCESS);
       exit(TEST_KO);

Diffs:
       atoi: |0|
    ft_atoi: |469762049|

ft_calloc:     [MISSING]
ft_strdup:     [MISSING]
ft_substr:     [MISSING]
ft_strjoin:    [MISSING]
ft_strtrim:    [MISSING]
ft_split:      [MISSING]
ft_itoa:       [MISSING]
ft_strmapi:    [MISSING]
ft_putchar_fd: [MISSING]
ft_putstr_fd:  [MISSING]
ft_putendl_fd: [MISSING]
ft_putnbr_fd:  [MISSING]
ft_lstnew:     [MISSING]
ft_lstadd_front: [MISSING]
ft_lstsize:    [MISSING]
ft_lstlast:    [MISSING]
ft_lstadd_back:[MISSING]
ft_lstdelone:  [MISSING]
ft_lstclear:   [MISSING]
ft_lstiter:    [MISSING]
ft_lstmap:     [MISSING]
ekeen@LAPTOP-88HC3NEP:~/42KL_Core
ekeen@LAPTOP-88HC3NEP:~/42KL_Core
ekeen@LAPTOP-88HC3NEP:~/42KL_Core/libft-unit-test$ man memchr
ekeen@LAPTOP-88HC3NEP:~/42KL_Core/libft-unit-test$ make f
 make -j 3 -C ../Libft
make[1]: Entering directory '/home/ekeen/42KL_Core/Libft'
cc -Wall -Wextra -Werror  -c -o ft_memchr.o ft_memchr.c
ft_memchr.c: In function 'ft_memchr':
ft_memchr.c:23:11: error: return discards 'const' qualifier from pointer target type [-Werror=discarded-qualifier
s]
   23 |    return (s);
      |           ^
cc1: all warnings being treated as errors
make[1]: *** [<builtin>: ft_memchr.o] Error 1
make[1]: Leaving directory '/home/ekeen/42KL_Core/Libft'
make: *** [Makefile:179: assets/libft.a] Error 2
```

9. In the scenario below, the sizeof operator below takes in a string literal "c". It outputs 2 bytes in the 2nd instance because c and null terminator for a string takes 2 bytes. The first example has 3 bytes: c character, space and null terminator. Interestingly, if I just put a character literal ('c'), it outputs 4 bytes because it's treated as an int due to ASCII conversion (I think).



main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
  char *ptr = "Hello";
  printf("%lu\n", sizeof("c "));
  printf("%lu\n", sizeof(int));
  printf("%lu\n", sizeof(float));
  printf("%lu", sizeof(double));
  return 0;
}
```

Console | Shell

```
clang-7 -pthread -lm -o main main.c
./main
3
4
4
8
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
    char *ptr = "Hello";
    printf("%lu\n", sizeof("c"));
    printf("%lu\n", sizeof(int));
    printf("%lu\n", sizeof(float));
    printf("%lu", sizeof(double));
    return 0;
}
```

Console Shell

```
clang-7 -pthread -lm -o main main.c
./main
2
4
4
8
```

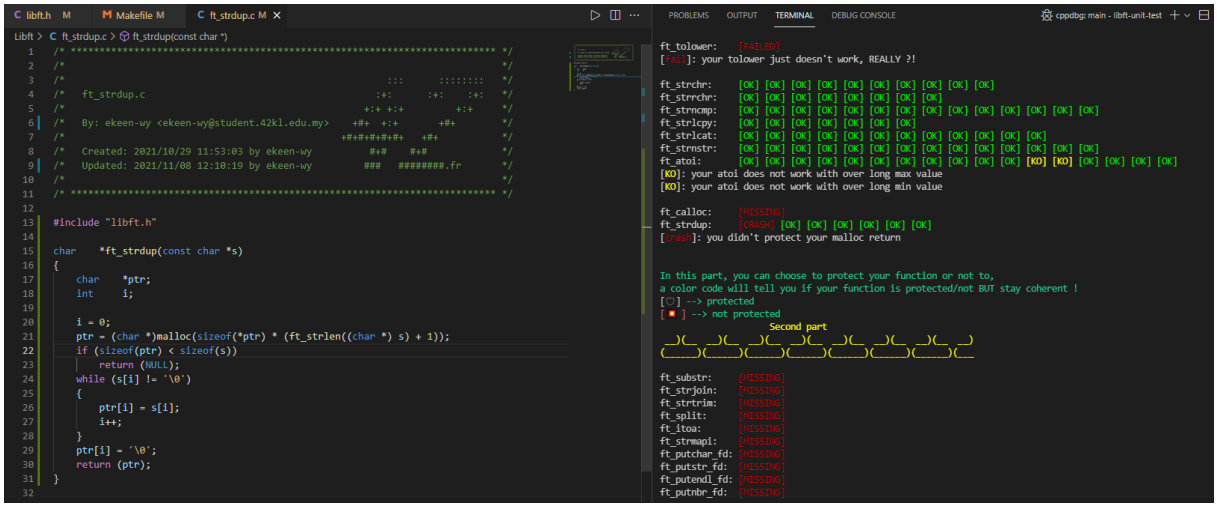
main.c

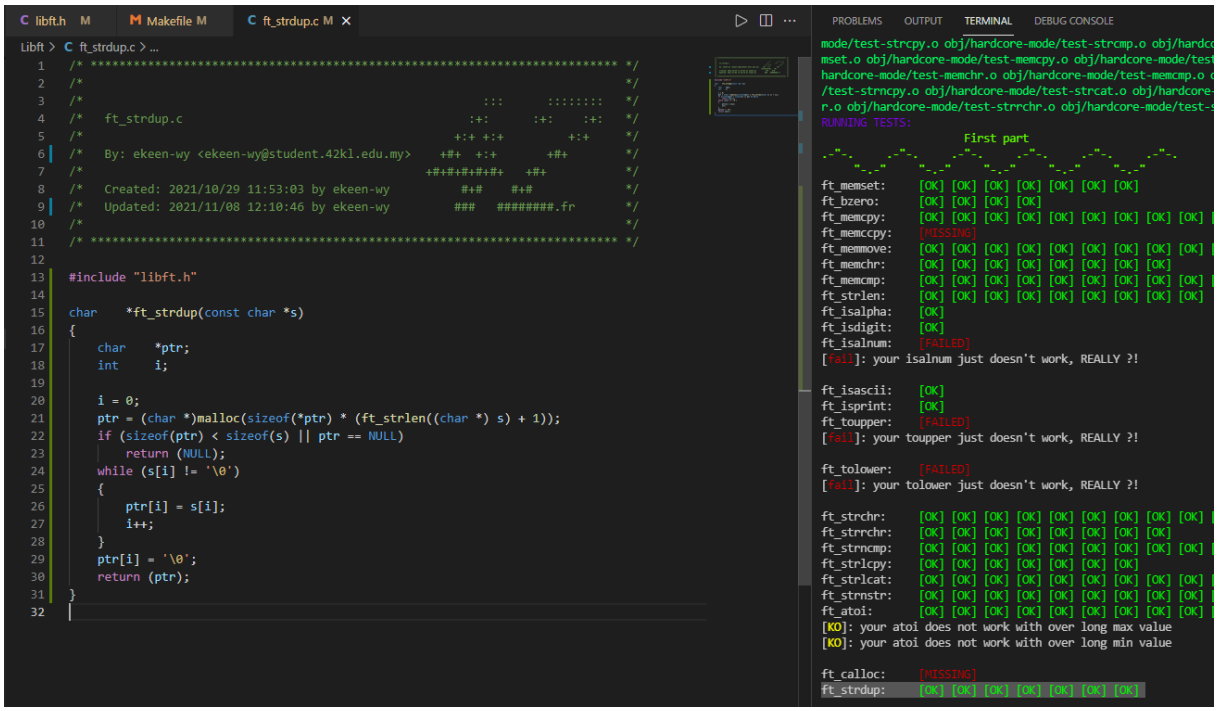
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
    char *ptr = "Hello";
    printf("%lu\n", sizeof('c'));
    printf("%lu\n", sizeof(int));
    printf("%lu\n", sizeof(float));
    printf("%lu", sizeof(double));
    return 0;
}
```

Console Shell

```
clang-7 -pthread -lm -o main main.c
./main
4
4
4
8
```

10. The strdup function in the first picture crashed because I didn't handle for the case where malloc fails to allocate any memory that was requested, which malloc will let the programmer know by returning NULL. Hence, by checking for this, I can protect the function.



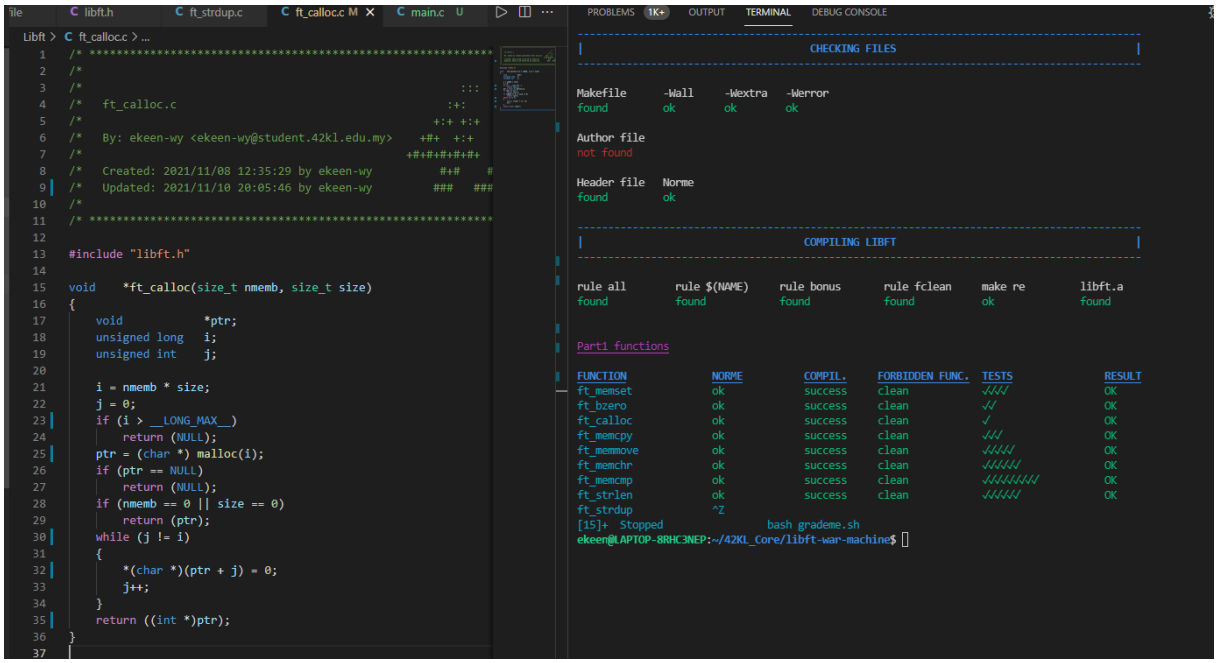


11. Why doesn't the compiler throw an error when typecasted my ptr return to a different type? From what I know about pointers so far, typecasting only works until the expression is evaluated by the compiler. It's still void in other context of the program. Is it because once I return, the function is poped of the stack, hence, the type cast disappears and isn't returned?

Void Pointers in C

Last updated on July 27, 2020 We have learned in chapter Pointer Basics in C that if a pointer is of type pointer to int or (int *) then it can hold the address of the variable of type int only.

https://overiq.com/c-programming-101/void-pointers-in-c/

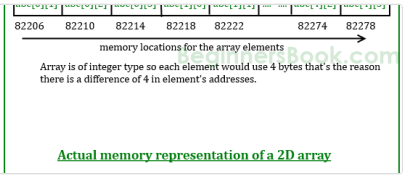


12. Array of Pointers vs 2-D Array vs Double Pointers

Two dimensional (2D) arrays in C programming with example

An array of arrays is known as 2D array. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns. Before we discuss more about two Dimensional array lets have a look at the following C program.

https://beginnersbook.com/2014/01/2d-arrays-in-c-example/



Why can't we use double pointer to represent two dimensional arrays?

Why can't we use double pointer to represent two dimensional arrays? arr[2][5] = {"hello","hai"}; **ptr = arr; Here why doesn't the double pointer (**ptr) work in this example?

https://stackoverflow.com/questions/4470950/why-cant-we-use-double-pointer-to-represent-two-dimensional-arrays

Double Pointer (Pointer to Pointer) in C - GeeksforGeeks

Prerequisite : Pointers in C and C++ We already know that a pointer points to a location in memory and thus used to store the address of variables. So, when we define a pointer to pointer. The first pointer is used to store the address of the variable.

https://www.geeksforgeeks.org/double-pointer-pointer-pointer-c/

Array of pointers to string in C Language | Codingeek

In this article we will discuss about the array of String pointers, what are it's benefits and drawbacks and we will implement the C program to understand these concepts in detail. contain addresses of the particular variable that we need.

https://www.codingeek.com/tutorials/c-programming/array-of-pointers-to-stringc-programming-language/

https://www.cs.cmu.edu/~ab/15-123S09/lectures/Lecture 04 - Pointers and Strings.pdf

13. I cannot write to the the location that the s pointer is pointing to because I haven't explicitly allocated memory for the string "guna" and string literal is stored in the read only section of memory.

https://www.geeksforgeeks.org/whats-difference-between-char-s-and-char-s-in-c/#:~:text=The string literal is stored,stores address of string literal.

Storage Classes in C - GeeksforGeeks

Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program. C language uses 4 storage classes, namely: auto: This is the default storage class for all the

https://www.geeksforgeeks.org/storage-classes-in-c/

How is read-only memory implemented in C?

As other folks have mentioned, whether the contents of constant strings are stored in read-only memory is determined by the operating system, compiler, and chip architecture. More precisely, the C standard specifies that the quoted strings are considered to have "const char[]" type (or words to that effect, I don't

https://stackoverflow.com/questions/1704893/how-is-read-only-memory-implemented-in-c

String literals: pointer vs. char array

Thanks for contributing an answer to Stack Overflow! Please be sure to answer the question. Provide details and share your research! Asking for help, clarification, or responding to other answers. Making statements based on opinion; back them up with references or personal experience. To learn more, see our tips on

https://stackoverflow.com/questions/12795850/string-literals-pointer-vs-char-array

What is the difference between char s[] and char *s?

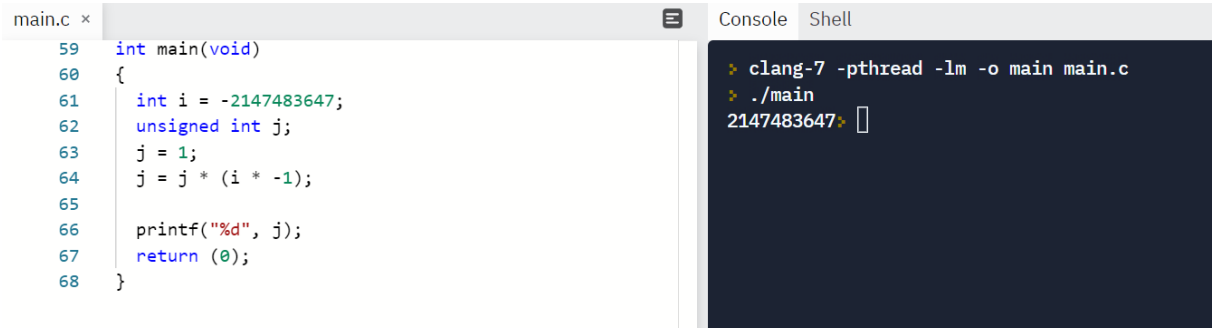
In C, one can use a string literal in a declaration like this: char s[] = "hello"; or like this: char *s = "hello"; So what is the difference? I want to know what actually happens in terms of s...

https://stackoverflow.com/questions/1704407/what-is-the-difference-between-char-s-and-char-s

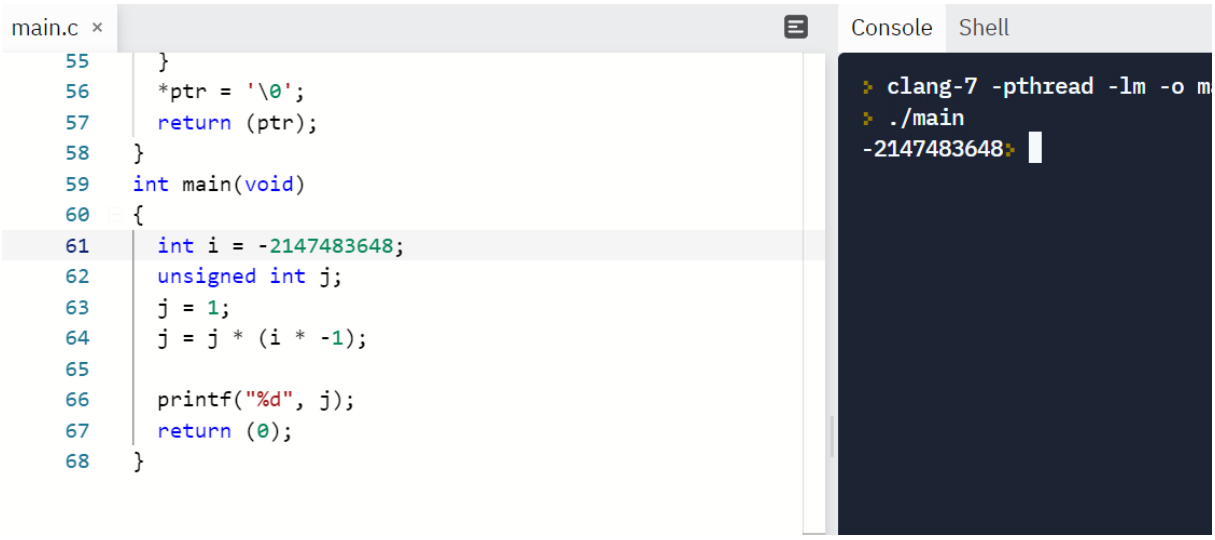

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>

int main(void)
{
  char *s1 = "   \t  \n\n \t\t  \n\n\nHello \t  Please\n Trim me !\n   \n \n \t\t\n  ";

  char *s = (char *)malloc(5);
  s = "guna";
  printf("%c", s[1]);

  return (0);
}
```

```
clang-7 -pthread
./main
u
```

14. Why when the int minimum, i can't just multiply it be -1 and assign in to unsigned int to get a positive number? Because of the formatter I used. If using %u then it's ok.




```c
int main(void)
{
  int i = -2147483647;
  unsigned int j;
  j = 1;
  j = j * (i * -1);

  printf("%d", j);
  return (0);
}
```



15. What are the risks of using unsigned int? Typecasting and integer promotion? How integer over and underflow are handled (INT MAX VS INT MIN)?

16. **Function Pointers**

Implementations of *ft_strmapi

```
char  *ft_strmapi(char const *s, char (*f)(unsigned int, char))
{
  unsigned int  len;
  unsigned int  i;
  char      *ptr;
  char      (*fptr)(unsigned int, char);

  if (s == NULL)
    return (NULL);
  len = ft_strlen((char *) s);
  ptr = (char *)malloc(sizeof(char) * (len + 1));
  fptr = &*f; // if I write in this manner, by putting an &, I am actually passing in the address of pointer pointing to the address of
  i = 0;
  if (ptr == NULL)
    return (NULL);
  while (len-- > 0)
  {
    *(ptr + i) = (*fptr)(i, s[i]);
    i++;
  }
  *(ptr + i) = '\0';
  return (ptr);
}
```

```
char  *ft_strmapi(char const *s, char (*f)(unsigned int, char))
{
  unsigned int  len;
  unsigned int  i;
  char      *ptr;
  char      (*fptr)(unsigned int, char);

  if (s == NULL)
    return (NULL);
  len = ft_strlen((char *) s);
  ptr = (char *)malloc(sizeof(char) * (len + 1));
  fptr = f;
  i = 0;
  if (ptr == NULL)
    return (NULL);
  while (len-- > 0)
```

```
  {
    *(ptr + i) = (*fptr)(i, s[i]);
    i++;
  }
  *(ptr + i) = '\0';
  return (ptr);
}
```

```
#include "libft.h"

char  *ft_strmapi(char const *s, char (*f)(unsigned int, char))
{
  unsigned int  len;
  unsigned int  i;
  char      *ptr;

  if (s == NULL)
    return (NULL);
  len = ft_strlen((char *) s);
  ptr = (char *)malloc(sizeof(char) * (len + 1));
  i = 0;
  if (ptr == NULL)
    return (NULL);
  while (len-- > 0)
  {
    *(ptr + i) = (*f)(i, s[i]);
    i++;
  }
  *(ptr + i) = '\0';
  return (ptr);
}
```

Function Pointers in C and C++

A function pointer is a variable that stores the address of a function that can later be called through that function pointer. This is useful because functions encapsulate behavior. For instance, every time you need a particular behavior such as drawing a line, instead of writing out a bunch of code, all you need to do is call

https://www.cprogramming.com/tutorial/function-pointers.html

How do you pass a function as a parameter in C?

A prototype for a function which takes a function parameter looks like the following: void func (void (*f)(int)); This states that the parameter f will be a pointer to a function which has a void return type and which takes a single int parameter.

https://stackoverflow.com/questions/9410/how-do-you-pass-a-function-as-a-parameter-in-c

What is the meaning of the declaration "char (* (*f())[])();"?

It breaks down as follows: f -- f f() -- is a function returning *f() -- a pointer to (*f())[] -- an array of *(*f())[] -- pointer to (*(*f())[])() -- function returning char (*(*f())[])(); -- char Postfix operators like [] and function call () bind before unary operators like *, so *a[] -- is an array of pointer (*a)[] -- is a pointer to an array *f() -- is a

https://stackoverflow.com/questions/25283815/what-is-the-meaning-of-the-declaration-char-f

17. Linked list implementations

```
#include "libft.h"

t_list  *ft_lstnew(void *content)
{
  t_list  *list;

  list = (void *)malloc(sizeof(t_list)); /* why does casting the malloc to void works? What does it mean to cast this to t_list?*/
  if (list == NULL)
    return (NULL);
  list -> content = content;
  list -> next = NULL;
```

```
  return (list);
}
```

```
#include "libft.h"

void  ft_lstclear(t_list **lst, void (*del)(void *))
{
  t_list  *tempnode;

  while (*lst != NULL)
  {
    tempnode = *lst;
    (*del)(tempnode -> content);
    *lst = **lst -> next; //error thrown: expression must have a pointer-to-struct-or-union type but it has type "t_list **".
    free(tempnode);
  }
  *lst = NULL;
}
```

18. Static librar

https://medium.com/@bdov_/https-medium-com-bdov-c-static-libraries-what-why-and-how-b6b442b054d3