# Homework 1

| | |
|---|---|
| Student Name | Muhammed Oğuz |
| Student Number | 1801042634 |

## Problem Understanding and Approach

First of all, I had to understand how can I use system calls and perform actions in C.

I rewatch lecture recordings. In lesson explanations were strictly enough to handle how can system calls used.

I determine that, I will only need to use `read`, `write`, `open` and `close` system calls.

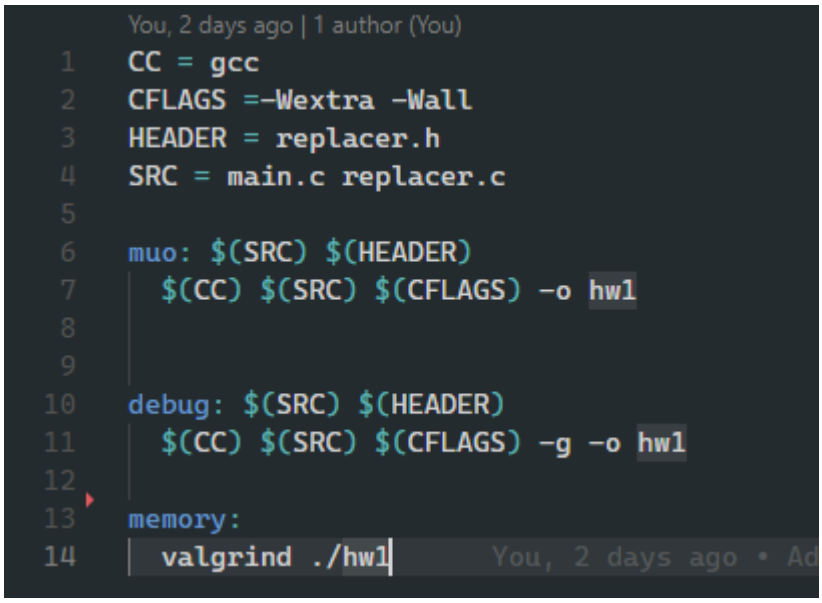
Apart from those system calls, since we are not allowed to use regex expressions, I will use a lot of C library functions and for loops.

## File Structure

`makefile`

I have a makefile that I using same structure for 2 years.

It basically contains variable to compile, extra flags and targets.



debug: This option will compile program with `-g` flag. It will allows you to debug. For entering debug mode, you have to run your program with `gdb`.

memory: This option will call `valgrind`. But since, We have to provide arguments, It has no effect for this homework. For valgrind, you have to run command without makefile.

### `replacer.h`

This file contains functions declarations. All functions has meaningful comments that contains, return value, brief explanation and params.

There is 3 type of function in my implementation.

main functions: Those functions are main functions. Also they will called from `main.c` file.

free functions: Those functions are used for free operations. With those functions, valgrind gives perfect result.

helper functions: Those functions are also important as main functions. But they are not called from main.

Also, there is 2 structure and 1 enum type in this file.

### ReplacePattern Structure

This structure holds information about send replace pattern. I used this as an array in my program. An array holds all possible pattern.

```
4   typedef struct
5   {
6     char *replace;
7     char *with;
8     int case_sensitive;
9     int match_multiple;
10    char *match_multiple_str;
11    int match_beginning;
12    int match_end;
13    int match_any;
14    char *match_any_str;
15  } ReplacePattern;
16
```

### Line Structure

This structure, holds a string array and a integer to hold a line and line count.

```
You, 20 hours ago | 1 author (You)
typedef struct
{
  char **words;
  int word_count;
} Line;
```

## Error Enum Type

This enum type holds all possible errors in my program. Also there is a function called `print_error_type(Error type)`, to show more information about the error.

```
typedef enum
{
  INVALID_MALLOC = -1,
  INVALID_MATCH_MULTIPLE = -2,
  INVALID_MATCH_ANY = -3,
  INVALID_MATCH_BEGINNING = -4,
  INVALID_MATCH_END = -5,
  INVALID_SLASH_COUNT = -6,
  INVALID_WORD_USAGE = -7,
  INVALID_ARGUMENTS = -8,
  INVALID_INITIALIZATION = -9,
  INVALID_CHAR_OCCURRENCE = -10,
  INVALID_REPLACE_PARAMETER = -11,
  INVALID_COMMA_USAGE = -12,
  FILE_OPEN_ERROR = -13,
  FILE_READ_ERROR = -14,
  FILE_WRITE_ERROR = -15,
  WORD_SPLIT_ERROR = -16,
} Error;
```

### `replacer.c`

This file contains implementation of `replacer.h` functions. It follows same order as `replacer.h`.

### `main.c`

This file holds main function to execute all functions.

## Test.txt

This text contains example for working with Homework PDF.

# Coding Approach

## Determine Patterns

First of all, I divide arguments and try to divide parts to meaningful data.

For achieving that, I used my `ReplacePattern` structure.

There is a function called `detect_replace_pattern`. This function detects replace pattern if all pattern are valid. If not, It will return a meaningful error code.

I tested almost every single possible. It works as expected.

Some test cases that I tested and works fine.

- `$` , `^` places are invalid ✔️
- `*` appears front ✔️
- `/` count does not satisfies the expected ✔️
- Given words are not valid (e.g: Not alphabetical) ✔️
- `;` Usage. (Next pattern is not send as expected) ✔️
- `[` and `]` are not match. ✔️

## Read and Write Operations

After determining all possible cases, I tested `read`, and `write` operations. I send some parameters and test strings. After achieving expected working mechanism, I those to functions.

## Split Read File

I use my `Line` struct to divide meaningful data of read file.

With power of this structure, Implementing replacement operations are way more easy.

## Check String Equality

There are a lot of little configuration to handle and check equality.

- Check if pattern has `$` or `^`.
- Check if pattern is incasesensitive.

- Check if pattern has `*`.
- Check if pattern has multiple choice. (`[` and `]`)

### Handling Memory

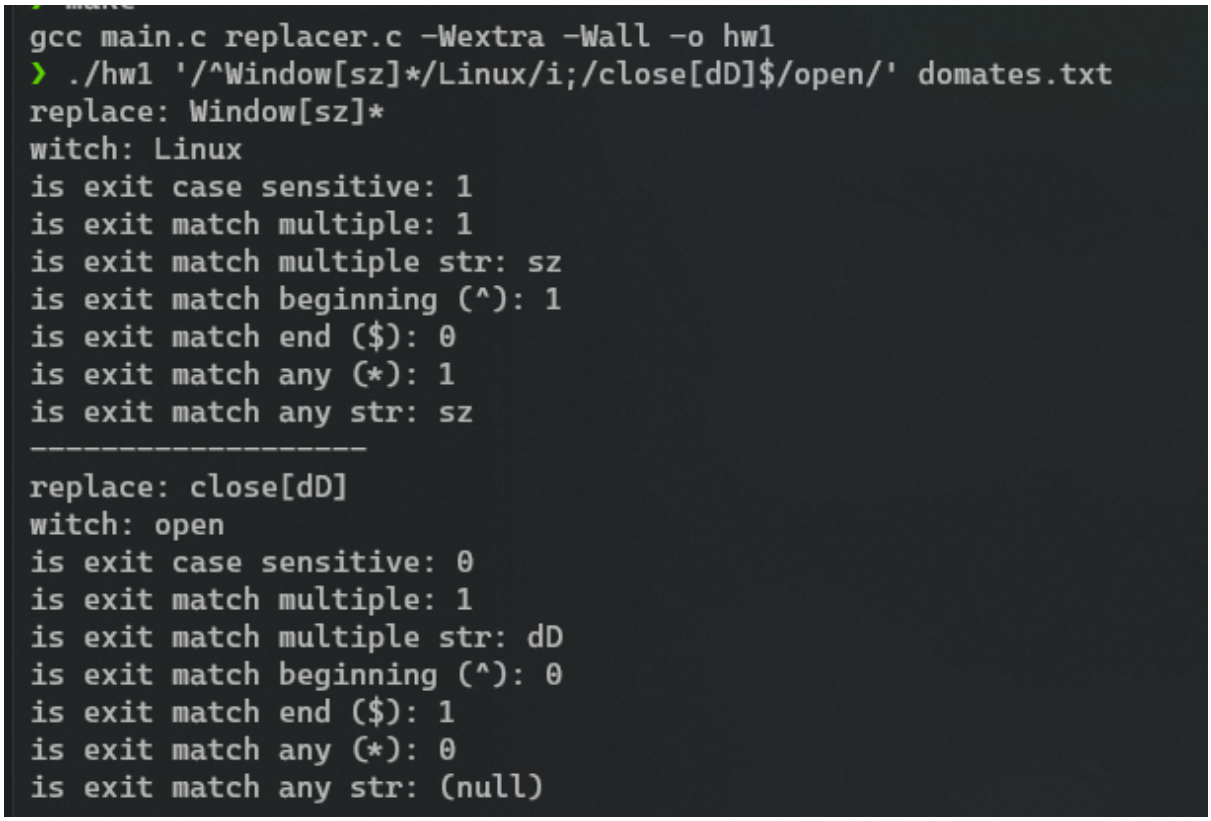
I wrote some free functions and I try to be careful about memory while implementing.

Most of the time I handled it. Some places were a bit challenging but I handled.

# Running and Outputs

## Detecting Patterns

> 💡 This example is commented in final code.

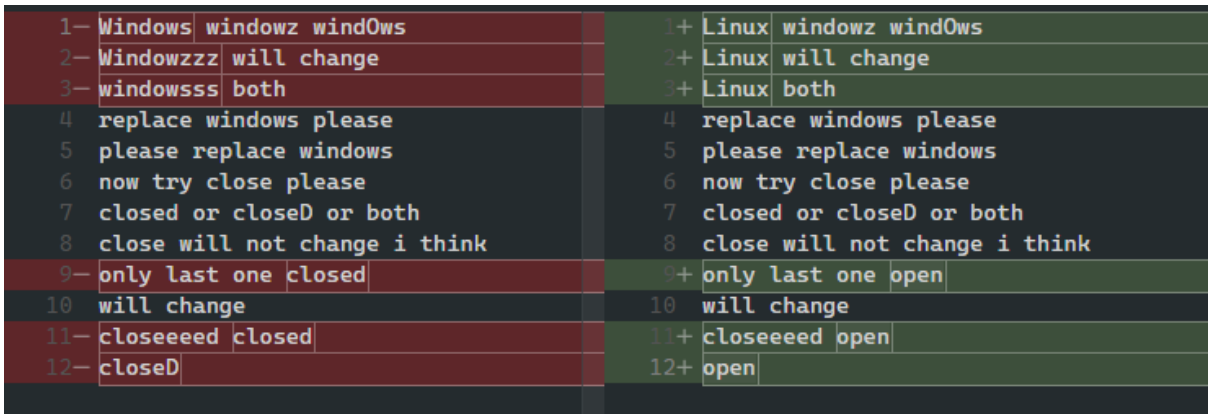



As shown above, It detects all expected pattern matchings.

## Testing Input in Homework PDF

```
./hw1 '/^Window[sz]*/Linux/i;/close[dD]$/open/' test.txt
```

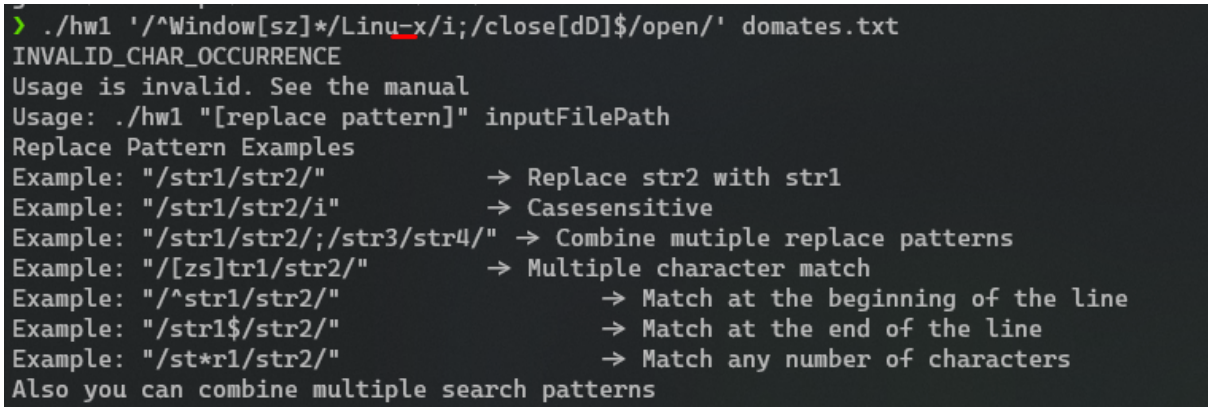
### Result




### Terminal Output of Successful Run


## Invalid Input Result

When user enters wrong argument, It will shows relative error, print manual and exit wit `EXIT_FAILURE`.




## Valgrind Result

### Normal

```
❯ valgrind ./hw1 '/^Window[sz]*/Linux/i;/close[dD]$/open/' domates.txt
==22313== Memcheck, a memory error detector
==22313== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22313== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==22313== Command: ./hw1 /^Window[sz]*/Linux/i;/close[dD]$/open/ domates.txt
==22313==
Successfully executed desired pattern
==22313==
==22313== HEAP SUMMARY:
==22313==     in use at exit: 0 bytes in 0 blocks
==22313==   total heap usage: 124 allocs, 124 frees, 2,250 bytes allocated
==22313==
==22313== All heap blocks were freed -- no leaks are possible
==22313==
==22313== For lists of detected and suppressed errors, rerun with: -s
==22313== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

### With `-leak-check=full` flag

```
❯ valgrind --leak-check=full ./hw1 '/^Window[sz]*/Linux/i;/close[dD]$/open/' domat
==22317== Memcheck, a memory error detector
==22317== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22317== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==22317== Command: ./hw1 /^Window[sz]*/Linux/i;/close[dD]$/open/ domates.txt
==22317==
Successfully executed desired pattern
==22317==
==22317== HEAP SUMMARY:
==22317==     in use at exit: 0 bytes in 0 blocks
==22317==   total heap usage: 124 allocs, 124 frees, 2,250 bytes allocated
==22317==
==22317== All heap blocks were freed -- no leaks are possible
==22317==
==22317== For lists of detected and suppressed errors, rerun with: -s
==22317== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

# Missing Parts

I checked almost everything. If I don't missed anything, I have only one failure for this homework.

When using `[abc]*` (using match multiple and match any). It works fine,

But since `*` symbol should behave, none or more occurrence.

If none occurence were applied, It does not working.

Example: /str[abc]*/change/ → straaa, strbb, strcccc will be change. But str will not work unfortunately.

But without combining them, `*` works fine in alone.

Example: /st*r/change/ → sttr, str, sr will be change.

I run so many tests and this is the only failure that I found myself.