Name: Fabiola INGABIRE

14th June 2024

Email:fabiolaingabire02@gmail.com

ROBOTIC TEAM

PURPOSE: Command-Line Argument Handling in C

Introduction

This report documents the development of a C program for handling command-line arguments, progressing from a basic "Hello, world!" program (`**helloworld.c**`) to an enhanced version (`**helloArgv.c**`) that supports variable outputs based on user-specified parameters.

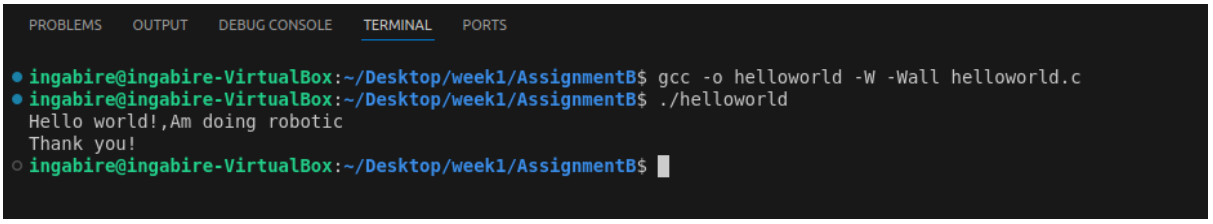
1. Initial Setup and Development

1.1 `helloworld.c`

The project commenced with the creation of `**helloworld.c**`, a simple C program that prints "Hello world! ,Am doing robotics" and other line prints to the console" Thank you!". This foundational program was designed to familiarise with basic C syntax and compiling processes.

1.2 Compilation and Execution

The program was compiled using the '**gcc**` compiler, which produced an executable file named `**helloworld**`. Execution of `**helloworld**` confirmed the correct functioning of basic print operations in C.



2. Project Expansion: `helloArgv.c`

2.1 Program Objective

Building upon **`helloworld.c`,** the project evolved into **`helloArgv.c`,** aimed at demonstrating command-line argument parsing capabilities in C. The enhanced program allows users to specify parameters (`--m` for morning messages and `--e` for evening messages) to control program output dynamically.

2.2 Development Process

2.2.1 Feature Implementation

- Argument Parsing: Implemented logic to parse command-line arguments (**`argv`)** using conditional checks and string comparisons (**`strcmp`**).

- Function Modularity: Introduced modular functions (`print_help`, `print_morning_message`, `print_evening_message`) to enhance code readability and maintainability.

2.3 Compilation and Execution

2.3.1 Compilation Steps

- The program was compiled using `gcc` with appropriate flags (`-o`) to generate an executable file (**`helloArgv`**).

2.3.2 Usage Scenarios

- Execution Examples : Demonstrated multiple usage scenarios, including printing morning messages (**`--m M`),** evening messages **(`--e E`)**, combined messages **(`--m M --e E`)**, and displaying help instructions **(`--help`).**

3. code in vs code:

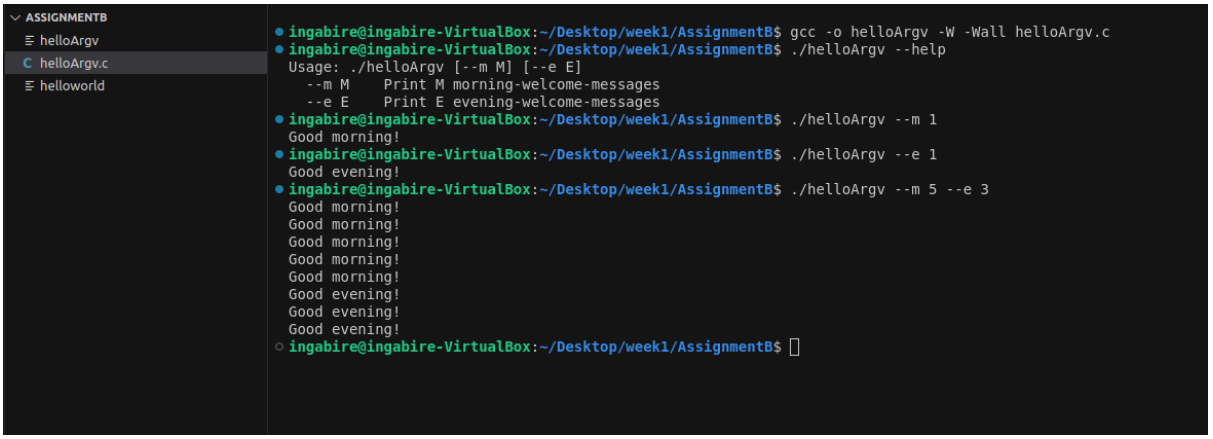
helloArgv.c > ...

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

void print_help() {
    printf("Usage: ./helloArgv [--m M] [--e E]\n");
    printf("  --m M    Print M morning-welcome-messages\n");
    printf("  --e E    Print E evening-welcome-messages\n");
}

void print_morning_message(int count) {
    for (int i = 0; i < count; i++) {
        printf("Good morning!\n");
    }
}

void print_evening_message(int count) {
    for (int i = 0; i < count; i++) {
        printf("Good evening!\n");
    }
}

int main(int argc, char *argv[]) {
    int morning_count = 0;
    int evening_count = 0;

    for (int i = 1; i < argc; i++) {
        if (strcmp(argv[i], "--help") == 0) {
            print_help();
            return 0;
        } else if (strcmp(argv[i], "--m") == 0) {
            if (i + 1 < argc && argv[i + 1][0] != '-') {
                morning_count = atoi(argv[++i]);
            } else {
                morning_count = 1;
            }
        } else if (strcmp(argv[i], "--e") == 0) {
            if (i + 1 < argc && argv[i + 1][0] != '-') {
                evening_count = atoi(argv[++i]);
            } else {
                evening_count = 1;
            }
```

```
38              if (i + 1 < argc && argv[i + 1][0] != '-') {
39                  evening_count = atoi(argv[++i]);
40              } else {
41                  evening_count = 1;
42              }
43          } else {
44              printf("Unknown option: %s\n", argv[i]);
45              print_help();
46              return 1;
47          }
48      }
49
50      if (morning_count > 0) {
51          print_morning_message(morning_count);
52      }
53
54      if (evening_count > 0) {
55          print_evening_message(evening_count);
56      }
57
58      return 0;
59  }
60
```

3.2 Sample Output

This document show the key command to run and debug our code and the output after typing the code in console



Conclusion

The evolution from `**helloworld.c**` to `**helloArgv.c**` illustrates a methodical progression in program refinement. This journey includes foundational learning, implementing new features, conducting thorough testing, and creating detailed documentation. It highlights the significance of adeptly managing command-line arguments in C programming, emphasizing adaptability, dependability, and user engagement as pivotal aspects of software development.