#### Command Line Arguments

Course: Introduction to Programming and Data Structures

#### Dr. Laltu Sardar

Institute for Advancing Intelligence (IAI), TCG Centres for Research and Education in Science and Technology (TCG Crest)





- Command-line Arguments in C
  - Syntax
  - Example program
  - Usage



# Command-line Arguments:

Input from terminal before execution



## Introduction to Command Line Arguments

- Another form of input
- Command-line arguments are inputs provided to a program at runtime.
- Useful when you want to control your program from outside.
- To override defaults and have more direct control over the application
- They allow users to interact with the program and control its behavior.



#### Syntax

```
int main(int argc, char *argv[]) {
    /* ... */
```

or

```
int main(int argc, char **argv) {
   /* ... */
```

- argc: Argument count the number of command-line arguments.
- argv: Argument vector an array of pointers to strings representing the arguments.
- argv[0] is the name of the program, After that till argv[argc-1] every element is command-line arguments.

Only strings can be taken from command line.

#### Understanding argc

- argc is an integer that represents the number of command-line arguments.
- The count includes the program name as the first argument.
- Example: ./program arg1 arg2 results in argc = 3.



#### Understanding argv

- argv is an array of character pointers (strings).
- argv[0] holds the name of the program.
- argv[1] to argv[argc-1] hold the subsequent arguments.
- Example:
  - argv[0] = "./program"
  - argv[1] = "arg1"
  - argv[2] = "arg2"



#### Example Program

```
#include <stdio.h>
int main(int argc, char * argv[]) {
    printf("Program name: %s\n", argv[0]);
    if (argc > 1) {
        for (int i = 1; i < argc; i++) {
            printf("Argument %d: %s\n", i, argv[i]);
        }
    } else {
        printf("No additional arguments passed.\n");
    }
    return 0;
}</pre>
```



#### Explanation of Example Program

- The program prints the name of the program and the arguments passed to it.
- If no additional arguments are passed, it informs the user.



## Another Example

```
1 // Program to compute average of two float variables
2 #include < stdio.h>
3 #include < stdlib.h > //that contains atof
5 float average(float a, float b){
      return ((a+b)/2.0);
 int main(int argc, char *argv[]){
      float a, b, avg;
      if (argc == 3) {
          a = atof(argv[1]); //converting string to float
          b = atof(argv[2]);
      }else{
13
          scanf("%f %f", &a, &b); // taking input from terminal
      avg = average(a, b); //Compauting avarage
      printf("%.2f",avg); //writing on terminal
      return 0:
```

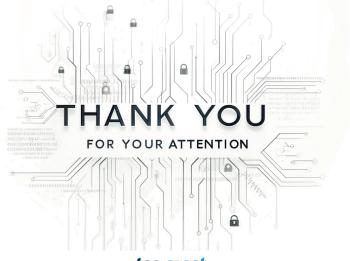
- Command-line arguments are often used to pass file names, options, and configurations.
- Common examples:
  - Example Programgcc myfile.c -o myfile
  - ./program input.txt output.txt



#### Handling Errors with Command Line Arguments

- Check argc to ensure the correct number of arguments is provided.
- Provide feedback to the user if required arguments are missing.





## tcg crest Inventing Harmonious Future

Dr. Laltu Sardar laltu.sardar@tcgcrest.org https://laltu-sardar.github.io.