In a Unix-like operating system (such as Linux or macOS), **ls** is a command used to list directory contents. It displays information about files and directories within the specified directory, or if no directory is specified, it lists the contents of the current directory.

* ***is: This command simply lists the names of files and directories in the current directory.***
* ***is -l: The -l option is used to list files in "long format." This format displays detailed information about each file, including permissions, ownership, size, and modification date.***
* ***is -l remind.c: This command lists detailed information about the file remind.c specifically. It will display information such as permissions, owner, group, size, and modification date/time for the file remind.c.***

int main(int argc, char \*argv[])

{

…

}

Argv—is the argument vector ,argc argument count

argv[0] points to the name of the program, while argv[1] through **argv[argc-1] point to the remaining command-line arguments.**

***argv[argc] is always a null pointer—a special pointer that points to nothing***.Note ,because the last character in the program points to nothing so its c,the second last character in a programm always points to the c-1

/\* Checks planet names \*/

#include<stdio.h>

#include<string.h>

#define NUM\_PLANETS 9

int main(int argc, char \*argv[])

{

char \*planets[] = {"Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"};

int i, j;

for (i = 1; i < argc; i++)

{

for (j = 0; j < NUM\_PLANETS; j++)

if (strcmp(argv[i], planets[j]) == 0)

{

printf("%s is planet %d\n", argv[i], j + 1);

break;

}

if (j == NUM\_PLANETS)

{

printf("%s is not a planet\n", argv[i]);

}

return 0;

}

1. **#include <stdio.h>** and **#include <string.h>**: These lines include the standard input/output and string manipulation libraries, respectively. They provide functions for printing to the console and comparing strings.
2. **#define NUM\_PLANETS 9**: This line defines a constant **NUM\_PLANETS** with a value of 9. This constant represents the number of planets in the **planets** array.
3. **int main(int argc, char \*argv[])**: This line declares the main function. It takes two arguments: **argc**, which stands for "argument count" and represents the number of command-line arguments passed to the program, and **argv**, which stands for "argument vector" and is an array of strings containing the command-line arguments themselves.
4. **char \*planets[] = {...}**: This line declares an array called **planets** containing pointers to strings. Each string represents the name of a planet in our solar system.
5. **int i, j;**: This line declares two integer variables **i** and **j**. These variables will be used as loop counters.
6. **for (i = 1; i < argc; i++) {**: This line starts a loop that iterates over each command-line argument passed to the program, starting from index 1 (**i = 1**). Index 0 (**argv[0]**) typically contains the name of the program itself.
7. **for (j = 0; j < NUM\_PLANETS; j++)**: This line starts a nested loop that iterates over each element of the **planets** array.
8. **if (strcmp(argv[i], planets[j]) == 0) {**: This line compares the current command-line argument (**argv[i]**) with each planet name (**planets[j]**) using the **strcmp** function. If the comparison returns 0, it means the strings are identical, indicating a match between the command-line argument and a planet name.
9. **printf("%s is planet %d\n", argv[i], j + 1);**: This line prints a message indicating that the current command-line argument is a planet name and its corresponding position in the **planets** array.
10. **break;**: This line exits the inner loop once a match is found, as there's no need to continue searching through the **planets** array.
11. **if (j == NUM\_PLANETS)**: This line checks whether the inner loop completed without finding a match for the current command-line argument.
12. **printf("%s is not a planet\n", argv[i]);**: If the current command-line argument doesn't match any planet name, this line prints a message indicating that it's not a planet.
13. **return 0;**: This line signifies the end of the **main** function and indicates successful program execution. It returns 0 to the operating system, indicating that the program terminated without errors.