3.7 Command Line Arguments

Objectives

- To understand how arguments can be passed to main function.
- To give input to the program when it is executed.
- To make program execution dynamic by changing input for each run.

Agenda

- Main Function
- Command-Line arguments
  - ✓ Types of arguments

- Example Programs using command-Line arguments
- Exercises

Main Function

- All C language programs must have a main() function.
- It's the core of every program.
- It contains instructions that tell the computer to carry out whatever task your program is designed to do.
- The main function can also have arguments

Command-Line arguments

- Arguments to the main function is called Command-Line arguments.
- A command-line argument is the information that follows the name of the program on the command line of the operating system.
- Command-line arguments are used to pass information into a program when the program is executed.
- Eg: When we write program to append two files ,the file names are supplied when program starts executing rather than specifying it as constants.

Introduction-Continued…

- C defines two built-in parameters to main()
  - The parameters receive the command line arguments
  - Their names are argc and argv

Note: The names of the parameters are arbitrary. However, argc and argv have been used by convention for several years.

Types of Parameters

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

argc:

- Holds the number of arguments on the command line
- Since the name of the program always corresponds to the first argument, it is always at least 1
- argc is an integer
- The value for this argument is not entered by the user.
- The system determined it from arguments that user specifies when program is executed.

Types of parameters-continued…

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

argv[]

- Argv is a pointer to an array of character pointers.
- Each character pointer in the argv array corresponds a string containing a command-line argument

  Eg: argv[0] points the name of the program, argv[1] points to the first argument, argv[2] points to the second argument, ...
- Each command-line argument is a string
- If you want to pass numerical information to your program, your program should convert the corresponding argument into its numerical equivalent.
- Each command-line argument must be separated by spaces or tabs

Syntax

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

When the program is executed:

In command prompt: $./a.out string1 string2…. stringN

Rules to be followed

- All command-line arguments are passed to the program as strings
  - program should convert them into their proper internal format.

- As a programmer, the names of the parameters in main can be specified, but the types and format are predefined for the language.

Illustrations

```
/*Program to print command-Line arguments*/
#include<stdio.h>
int main(int argc,char * argv[])
{
int i;
printf(“Number of arguments is:%d\n",argc);
printf(“Name of the program is :%s\n",argv[0]);
for(i=1;i<argc;i++)
{
printf(“User entered string value no %d is %s\n",i,argv[i]);
}
}
```

Output:

[d_bharathi@ssh ~]$ cc command1.c
[d_bharathi@ssh ~]$ ./a.out welcome
No of arguments is :2
Name of the program is:./a.out
User entered string value no 1 is welcome

```
/*Program to add two numbers*/
#include<stdio.h>
void main(int argc, char * argv[]) {
  int i, sum = 0;

  if (argc != 3) {
    printf("You have forgot to type numbers.");
    exit(1);
  }
  printf("The sum is : ");
  for (i = 1; i < argc; i++)
    sum = sum + atoi(argv[i]);
   printf("%d", sum);}
```

Output:

[d_bharathi@ssh ~]$ cc addcommand.c
[d_bharathi@ssh ~]$ ./a.out 5 7
The sum is : 12

Illustration with files-Program to copy one file content to another file

```
/* File Copy using Command line arguments */

#include<stdio.h>
int main(int argc,char *argv[])
{
 FILE *fs,*ft;
 int ch;
 if(argc!=3)
 {
  printf("Invalide numbers of arguments.");
  return 1;
 }
 fs=fopen(argv[1],"r");
 if(fs==NULL)
 {
  printf("Can't find the source file.");
  return 1;
 }
 ft=fopen(argv[2],"w");
 if(ft==NULL)
 {
  printf("Can't open target file.");
  fclose(fs);
  return 1;
 }

 while(1)
 {
  ch=fgetc(fs);
  if (feof(fs)) break;
  fputc(ch,ft);
 }

 fclose(fs);
 fclose(ft);
 return 0;
}
```

Illustration with files-Program to copy one file content to another file-continued…

Output:

[d_bharathi@ssh ~]$ vi con.c

[d_bharathi@ssh ~]$ vi sample.txt

[d_bharathi@ssh ~]$ vi result.txt

[d_bharathi@ssh ~]$ cc con.c

[d_bharathi@ssh ~]$ ./a.out sample.txt result.txt

[d_bharathi@ssh ~]$ vi result.txt

Finding the output

1. What will be the output of the program (myprog.c) given below if it is executed from the command line?

cmd> *myprog one two three*

```
/* myprog.c */
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
 { printf("%s\n", argv[1]);
 return 0; }
```

Output:

one

Finding the output-continued…

2. What will be the output of the program (sample.c) given below if it is executed from the command line (turbo c under DOS)?

cmd> *sample Good Morning*

```
/* sample.c */
#include<stdio.h>
int main(int argc, char *argv[])
{ printf("%d %s", argc, argv[1]); return 0; }
```

Output: 3 Good

Debugging code

1. What will be the output of the program (sample.c) given below if it is executed from the command line ?

cmd> *sample 1 2 3*

```
/* sample.c */
#include<stdio.h>
 int main(int argc, char *argv[])
 { int j; j = argv[1] + argv[2] + argv[3];
 printf("%d", j); return 0; }
```

Output:Error

Explanation: Here *argv[1]*, *argv[2]* and *argv[3]* are string type. We have to convert the string to integer type before perform arithmetic operation.

Example: *j = atoi(argv[1]) + atoi(argv[2]) + atoi(argv[3]);*

Simple word problems

1. Every time we supply new set of values to the program at command prompt, we need to recompile the program.

Answer: No only input will be changed.

2. The first argument to be supplied at command-line must always be count of total arguments

Answer: No, The system determined it from arguments that user specifies when program is executed.

Summary

- The Command-Line arguments provides input to the program during run time.
- Command-Line arguments are optional.