CL1002 – Programming Fundamentals Lab

Lab # 14

Pointers & File Handling

Instructor: Engr. Muhammad Usman

Email: usman.rafiq@nu.edu.pk

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar

C Pointers

Pointers are powerful features of C and C++ programming. Before we learn pointers, let's learn about addresses in C programming.

Address in C

If you have a variable var in your program, &var will give you its address in the memory.

We have used address numerous times while using the scanf() function.

```
scanf("%d", &var);
```

Here, the value entered by the user is stored in the address of var variable. Let's take a working example.

Example 1 | Address in C

```
#include <stdio.h>
int main()
{
 int var = 5;
 printf("var: %d\n", var);

 // Notice the use of & before var
 printf("address of var: %p", &var);
 return 0;
}
```

Output

```
var: 5
address of var: 0x7ffeb9656ca4
```

Note: You will probably get a different address when you run the above code.

C Pointers

Pointers (pointer variables) are special variables that are used to store addresses rather than values.

Pointer Syntax

Here is how we can declare pointers.

```
int* p;
```

Here, we have declared a pointer *p* of *int* type.

Assigning addresses to Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
```

Here, 5 is assigned to the *c* variable. And, the address of *c* is assigned to the *pc* pointer.

Get Value of Thing Pointed by Pointers

To get the value of the thing pointed by the pointers, we use the * operator. For example:

```
int* pc, c;
c = 5;
pc = &c;
printf("%d", *pc);   // Output: 5
```

Here, the address of c is assigned to the *pc* pointer. To get the value stored in that address, we used **pc*.

Changing Value Pointed by Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Output: 1
printf("%d", *pc);  // Ouptut: 1
```

We have assigned the address of *c* to the *pc* pointer.

Then, we changed the value of *c* to 1. Since *pc* and the address of *c* is the same, **pc* gives us 1.

Let's take another example.

```
int* pc, c;
c = 5;
pc = &c;
```

```
*pc = 1;
printf("%d", *pc);   // Ouptut: 1
printf("%d", c);     // Output: 1
```

We have assigned the address of *c* to the pc pointer.

Then, we changed **pc* to 1 using **pc = 1;*. Since *pc* and the address of *c* is the same, *c* will be equal to 1.

Example 2 | Working of Pointers

```
#include <stdio.h>
int main()
{
  int* pc, c;

  c = 22;
  printf("Address of c: %p\n", &c);
  printf("Value of c: %d\n\n", c);  // 22

  pc = &c;
  printf("Address of pointer pc: %p\n", pc);
  printf("Content of pointer pc: %d\n\n", *pc); // 22

  return 0;
}
```

Output

```
Address of c: 0x7fff17a620fc
Value of c: 22

Address of pointer pc: 0x7fff17a620fc
Content of pointer pc: 22
```

Example 3 | Write a program in C to add two numbers using pointers.

```
#include <stdio.h>
int main()
{
```

```c
  int fno, sno, *ptr1, *ptr2, sum;

  printf("\nPointer : Add two numbers :\n");
  printf("-------------------------------\n");

  ptr1 = &fno;
  ptr2 = &sno;

  printf(" Input the first number : ");
  scanf("%d", ptr1);
  printf(" Input the second  number : ");
  scanf("%d", ptr2);
  sum = *ptr1 + *ptr2;
  printf(" The sum of the entered numbers is : %d\n\n",sum);
  return 0;
}
```

Output

```
Pointer : Add two numbers :
-------------------------------
 Input the first number : 3
 Input the second  number : 4
 The sum of the entered numbers is : 7
```

C File Handling

A file is a container in computer storage devices used for storing data.

So far the operations using C program are done on a prompt / terminal which is not stored anywhere. But in the software industry, most of the programs are written to store the information fetched from the program. One such way is to store the fetched information in a file.

Why files are needed?

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

- You can easily move your data from one computer to another without any changes.

Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

1. Text files

Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad.
When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.
They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

```
FILE *fptr;
```

Opening a file - for creation and edit

Opening a file is performed using the fopen() function defined in the stdio.h header file.
The syntax for opening a file in standard I/O is:

```
ptr = fopen("fileopen","mode");
```

For example,

```
fopen("/home/usman/pf/program.txt","w");
```

- Let's suppose the file newprogram.txt doesn't exist in the location `/home/usman/pf`. The first function creates a new file named *program.txt* and opens it for writing as per the mode 'w'.
  The writing mode allows you to create and edit (overwrite) the contents of the file.

Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. Data is added to the end of the file.	If the file does not exist, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exist, it will be created.

Closing a File

The file should be closed after reading/writing.

Closing a file is performed using the *fclose()* function.

```
fclose(fptr)
```

Here, fptr is a file pointer associated with the file to be closed.

Reading and writing to a text file

For reading and writing to a text file, we use the functions *fprintf()* and *fscanf()*.
They are just the file versions of *printf()* and *scanf()*. The only difference is that *fprintf()* and *fscanf()* expects a pointer to the structure FILE.

Example 4 | Write to a text file

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num;
  FILE *fptr;

// use appropriate location if you are using windows i.e (C:\\program.txt)
  fptr = fopen("/home/usman/pf/program.txt","w");
  if(fptr == NULL)
  {
     printf("Error!");
     exit(1);
  }
  printf("Enter num: ");
  scanf("%d",&num);

  fprintf(fptr,"%d",num);
  fclose(fptr);
  return 0;
}
```

This program takes a number from the user and stores in the file program.txt.
After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

Example 5 | Read from a text file

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
  char num[10];
  FILE *fptr;
// use appropriate location if you are using windows i.e
(C:\\program.txt)
  if ((fptr = fopen("/home/usman/pf/program.txt","r")) == NULL){
      printf("Error! opening file");
      // Program exits if the file pointer returns NULL.
      exit(1);
  }
  fgets(num,sizeof(num),fptr);
 //alternatively, we could also use
 // fscanf(FILE *ptr, const char *format, ...) for reading

  printf("Value of n=%s", num);
  fclose(fptr);
  return 0;
}
```

This program reads the integer present in the program.txt file and prints it onto the screen.

If you successfully created the file from Example 1, running this program will get you the integer you entered.

References:

https://www.programiz.com/c-programming/c-pointers

https://www.programiz.com/c-programming/c-file-input-output