Pointer 2

Computer Science Department

Eastern Washington University

Yun Tian (Tony) Ph.D.

Recall

- Concept of address
- Concept of pointers
- How to define/declare pointers
- How to use pointers?

Today

- Pointer Arithmetic

NULL

- It is always a good practice to assign a NULL value to a pointer variable in case you do not have exact address to be assigned.
  - This is done at the time of variable declaration.
- A pointer that is assigned NULL is called a null pointer.
  - NULL is predefined Constant in C standard library. The value of NULL is 0.

NULL

```
#include <stdio.h>

int main ()
{
   int  *ptr = NULL;

   printf("The value of ptr is : %p\n", ptr  );

   return 0;
}
//output The value of ptr is 0
```

Intro to pointer Arithmetic

- C pointer variable holds an address of another regular variable.
  - The address is a numeric value.
- You can perform
  - increase the pointer,
  - decrease the pointer,
  - or compare the value in a pointer variable with another address.

Intro

- These operations are called pointer arithmetic.
- Pointer arithmetic changes the value in the pointer variable,
  - In turn, these pointer arithmetic make the pointer point to **different** memory locations.

Example

- int a[4] = { 8, 9, 5, 6};
- int *ptr = a;
- In this example, we assume &a[0] is 0x1000, the base address of the array **a**.
- We assume each integer take 4 bytes on this machine.
- if we do ptr ++;
- what is the value of ptr now?

0x1004

Example

- int a[4] = { 8, 9, 5, 6};
- int *ptr = a;
- ptr ++
- ptr get 0x1004, **not 0x1001**.
- Because each time ptr is incremented, it will point to the next **integer** location which is 4 bytes next to the current location.
  - Because type of **ptr** is **a integer pointer**,(int *ptr).
  - Supposed to point to an integer.

Example

- int a[4] = { 8, 9, 5, 6};
- int *ptr = a;
- ptr ++
- ptr gets 0x1004, **not 0x1001**.
- In this example, 0x1004 is the address of a[1].
- You can understand this way:
  - initially ptr points a[0], because ptr = a,
  - after ptr ++, ptr points to the next integer memory location in a. that means it points to a[1].

0x1000	0x1004	0x1008	0x100c
a[0]	a[1]	a[2]	a[3]

Example

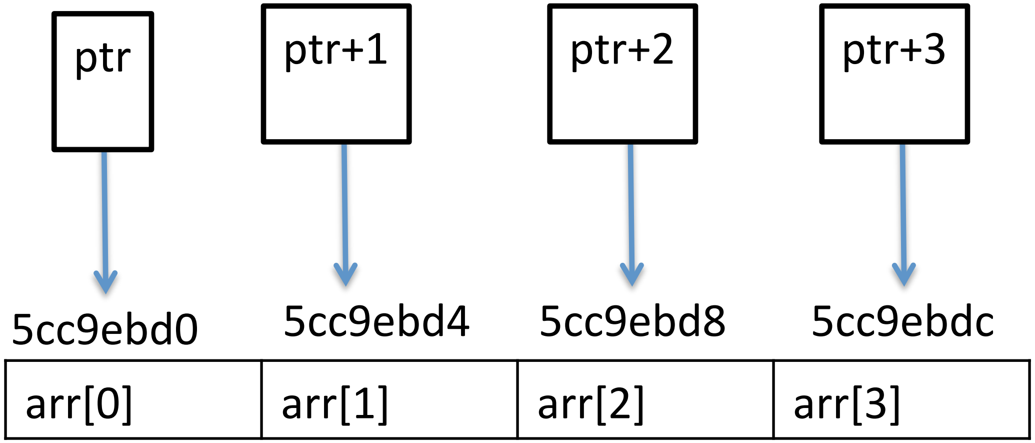
- This operation will move the pointer to next memory location of the data **item** without impacting actual value at the memory location.
- If **ptr** points to a **character** array whose address is 1000, then above operation will point to the location 1001
  - because next character will be available at 1001.
  - each character take one bytes in memory.

Demo of Pointe Arithmetic

- Summary
- ptr ++
- ptr = ptr + 1
- ++ ptr
  - ptr is increased by the size of data type it points to.
  - move the pointer to the next Mem location of the data type it points to.

Access Array Element

- int arr[] = { 10, 20, 30, 40};
- int *ptr = arr;



ptr
ptr+1
ptr+2
ptr+3

5cc9ebd0	5cc9ebd4	5cc9ebd8	5cc9ebdc
arr[0]	arr[1]	arr[2]	arr[3]

Then we could access each array element by use ***(ptr ++) or *(ptr + i) in a loop,**

Access of Array Element 1

```
int  i;
 for ( i = 0; i < MAX; i++)
 {
     //Assume ptr has been initialized to the base address of an array
     printf("Address of arr[%d] = %x\n", i, ptr + i );
     printf("Value of arr[%d] = %d\n", i, *(ptr  + i) );
 }
```

Access of Array Element 2

```
int  i;
 for ( i = 0; i < MAX; i++)
 {
     //Assume ptr has been initialized to the base address of an array
     printf("Address of arr[%d] = %x\n", i, ptr );
     printf("Value of arr[%d] = %d\n", i, *(ptr  ++) );
 }
```

Demo of Pointer Decreasing and Comparison

- pointerArith2.c
- pointerArith3.c

Summary

- Pointer Arithmetic
  - Increasing
  - Decreasing
  - Compare
  - Demo

Next Class

- Make files and Debug tools