Introduction to Pointers

Effective Programming in C and Unix

Announcements
- SL3 is due Thursday 1/27 midnight
- Complete the Academic Honesty Form
  - Link available from Bb->Course information
  - Your current courses grades are available from
    - /afs/andrew/course/15/233/grades
  - All SL and lab feedback are given at
    - /afs/andrew/course/15/233/handback

Learning Objectives
- Review of hexadecimal number system
- Understand how pointers work
- Understand how to access memory using pointers
- Understand pointer arithmetic
- Understand relation between arrays and pointers
- Understand the dangers of indirect memory access

Hexadecimals revisited
- Hexadecimal number system is a convenient way to represent binary form of data or addresses
- Hexadecimal digits vary from 0,1,...,9, a,b,c,d,e,f
- Each one is defined using 4 bits

| 1 | 0 | 1 | 1 |

- Hexadecimal addition can be performed using base-16 arithmetic. That is \( m = n \mod(16) \)

Definition of a pointer
- A pointer is an address of a memory location
- Address of a variable is called the lvalue

```
int x = 10
```

- lvalue of x =
- rvalue of x =

Pointer variables
- `<type>`* `variable_name`;
  - `int`* `ptr`; `char`* `ptr`;

- Initializing
  - `int`* `ptr` = &x;

- Dereferencing
  - `*ptr` = x;
Pointer Arithmetic
- <type>* ptr;
- ptr ➔ address of a memory location
- ptr + 1 ➔ address of the next <type> variable
- ptr1 - ptr2 ➔ number of type variables between ptr1 and ptr2

Question
- Given the address of an int variable x
  - ptr = &x = 0xFFA7B8C9
  - Find the address of the next int the memory (assuming it exists)
  - Find the address of the second byte of x (assuming x is a 32-bit int)

Accessing Memory using pointers
- Suppose an int is defined as ➔ int x = 10
- How do we print the address of x?
- How do we access the first byte of x?
- How do we access the second byte of x?
- How do we access the next 4 bytes after x? What are the dangers?
- Code demos

Arrays
- Defining an array
  - int A[10]
  - char* A[10]
  - int* A[10]
- Array Memory allocation
  - Allocates a Contiguous block of memory
  - The name of the array A is a constant pointer to the first element of the array
  - int A[10];
  - printf("%x", A);
- Dangers
  - C Arrays are not bounded. That is, one can violate memory not allocated using pointers

Array as a pointer

Calculate the addresses of each element

Understanding Arrays
- A ➔ address of the first element of the array
- A + i = address of A[i] = &A[i]
Dynamic Arrays

- int* A = (int*)malloc(n*sizeof(int));
- for (i=0; i<n; i++)
  A[i] = i;
- Resizing an array

More about Strings

- What is the difference between
  - char word[10]
  - char* word
- Look at the size of each of the above

Which of the following code seg faults? Explain...

- Assume we declare
  - char* word; char word2[10];
- Consider the following
  - strcpy(word, "guna");
  - strcpy(word2, "guna");
  - word = "guna";
  - Word2 = "guna";

Dangerous code

```c
int* foo(int n) {
    int x = n*n;
    return x;
}
```

```c
int* foo(int n) {
    int x = n*n;
    return &x;
}
```

C programming – important

- include stdlib.h in all your programs
- Declare functions and variables before using them
- increment and decrement with ++ and -- operators.
- Use x = 5 instead of x = x * 5
- A string is an array of characters ending with a "\0".
- Don’t ever forget the null character
- Array of size n has indices from 0 to n-1. Although C will allow you to access A[5], it is very dangerous...
  - A character can be represented by an integer (ASCII value) and can be used as such.
  - The unary operator & produces an address
  - The unary operator * dereferences a pointer
  - Arguments to functions are always passed by value. But the argument can be an address of just a value
  - For efficiency, pointers can be passed to or return from a function.
  - Logical false is zero and anything else is true
  - You can do things like for( ) or while( ) for program efficiency and writability
  - Use "" instead of "\"
- Always compile your program with -ansi -pedantic -Wall -std=99 flags

Coding Examples