Data Representations

15-123
Effective Programming in C and Unix

Learning Objectives

- At the end of this lecture, you should be able to
  - Understand how data is represented
  - Understand how integers are represented
  - Understand how negative numbers are represented
  - Understand how computers add numbers
  - Understand logical operations

Few reminders and demos

When there is a discrepancy between schedule and the date on the assignment go with assignment date.

Always ask when in doubt

Assignments are officially released within one week from the due date. Although you may be able to find a draft on download site, be sure to check back or talk to the course staff about changes

Getting Started

- Install SSH on your computer (and FTP)
- Learn how to use Unix on Gates and Wean Clusters
- In order to be successful in this course
  - You must be committed
  - You must work hard
  - You must seek help when needed
- Demos
  - Create sub folders for program development
  - Edit the program with emacs or equivalent
  - Test, debug and submit to handin

Salon - a new concept on social learning

- Make sure you have an account on Salon
  - If not use code: 23456 to generate one or send email
- Join salons 15-123S11
- Complete "prior knowledge" salon
  - Vote for good responses
- Join your section Salon for rest of the semester activities
- Your course will have (alternatively)
  - Salons and Quizzes
  - Salons are collaborative – Graded
  - Quizzes are the good old take home quizzes

Lab Assignments in this course

- Requires you to log your activities. Labs must be completed on unix (no IDE's). Before any active session do the following
  - > script session1.txt
  - > your session commands
  - > cntrl D (saves the session)
  - This is required for each lab (3 points)
  - Also manually log your times
    - Follow the formats carefully
    - These files will be automatically processed

Plan to attend help sessions this weekend

You need to start right

Start reading lab 1
you need more concepts from next week lectures

How data is represented

Data and Instructions

- Computers deal with two things
  - Data
    - int x = 10
  - Instructions
    - add x,y
    - store result, z

- Data as instructions and instructions as data
  - Both Data and Instructions are represented the same way

Representing Information

- Smallest Data unit is the *"bit"*
- Smallest addressable unit is the *"byte"*
- Each computer has a *"word"* size
  - "word" is the amount of memory transferred between CPU and RAM
  - Indicate the nominal size of integers and pointers
  - Most common size is 32-bits
    - 32-bit instructions, 32-bit integers
- Based on word size we can determine addressable space

Terminology

- 1 byte = 8 bits
- 1000 bytes = 1 kilo byte = 1 kb
- 1,000,000 bytes = 1 mega byte = 1 MB
- 1,000,000,000 bytes = 1 giga byte = 1 GB
- 1,000,000,000,000 bytes = 1 tera byte = 1 TB
- 1000 PB's = 1 peta byte = 1 PB
- Then we have exabyte, zettabyte and yotabyte

Question

- If a computer has 32-bit word size, what would be the range of virtual address space or max RAM?
- What if the computer is a "64-bit" machine?

Data Sizes

- Here are the typical 32-bit allocation for data types (in bytes)
  - char (1), short int (2), int (4), long int (4)
    - In compaq alpha long int is 8
  - char* (4), float (4), double (8)
- The exact size of data allocation depends on both compiler and machine

Data value ranges

- <limits.h> library defines the range of values any data type can assume.
- Applications that are designed to be portable must use symbolic constants.
- **Some examples**
  - **INT_MIN**
    - Minimum value for a variable of type **int**.
    - -2147483647 - 1
  - **INT_MAX**
    - Maximum value for a variable of type **int**.
    - 2147483647
  - **UINT_MAX**
    - Maximum value for a variable of type **unsigned int**.
    - 4294967295 (0xffffffff)
  - **LONG_MIN**
    - Minimum value for a variable of type **long**.
    - -2147483647 - 1
  - **LONG_MAX**
    - Maximum value for a variable of type **long**.

Data Storage Classes

- auto
  - Typical variables defined inside functions
- static
  - Variables that retain values between function calls
- extern
  - Declared within a function, but specifications given else where
- Register
  - Data allocated to a register for quick access

How integers are represented

Integer Representations

- Typical 32-bit machine uses
  - 32-bit representation for signed and unsigned ints
    - Range:
  - Compaq alpha uses 64 bits for long int
    - Range:

Representation formats

- Decimal
- Binary $b_{n-1}\ldots b_1 b_0{}_{[2]} = b_{n-1}2^{n-1} + \cdots + b_1 2^1 + b_0 2^0 = \sum_{i=0}^{n-1} b_i 2^i$
- Octal
- Hexadecimal

Class/home work

- Convert 526 to binary, octal and hexadecimal
- /* prints the binary representation of a given int */

```
void printBinary(int n) {


}
```

Questions

- Suppose we write
  - int x = oxoF // to represent 15
- How would you write
  - The largest 32-bit positive integer
  - The 6th power of 2
  - A number that is divisible by 2

Addressing and byte ordering

- Little Endian
  - Least significant byte first (DEC, Intel)
- Big Endian
  - Most significant byte first (IBM, Motorola, SUN)
- Application programmers may not care about this ordering

When byte ordering becomes an issue

- Communication of binary data over a network between different machines
- Code written for networking applications must then do their own conversions between machines

How about negative integers?

Representing Negative numbers

- The definition of –x
  - -x is the number y such that: x + y = 0
  - y is called the additive inverse
- Consider the 4-bit addition of signed numbers

$$\begin{array}{cccccc} & 0 & 1 & 1 & 1 \\ + & 1_1 & 0_1 & 0_1 & 1 \\ \hline (1) & 0 & 0 & 0 & 0 \end{array}$$

Class work

- Add the following using binary addition
- 00001011 and 11110101

How are signed and unsigned integers represented?

- Consider a n-bit integer representation of an unsigned integer
- Consider a n-bit integer representation of a signed integer

Representing negative numbers using 2's complement

- One's complement
  ~x
- Two's complement
  1 + ~x

Signed and unsigned numbers

- By default all constant values are signed
  - const int x = 20, y =0x45
- Can create an unsigned constant using
  - const unsigned x = 0x123u (or U)

Performing logical operations

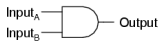
Logical Operations in C

- Logical AND (&&)
  - 0x75 && 0x 96
- Logical OR (||)
- Logical Not (!)

Bit Operations in C

- Bitwise AND (&)
  - 0X75 & 0X96

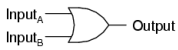
AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

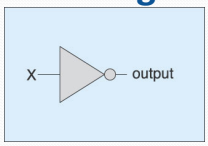
Bitwise OR (|)

2-input OR gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise negation (~)



XOR (^)

Exclusive-OR gate

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Logic for adding bit by bit

- Si = (Ai ^ Bi) ^ Cin
- Cout = (Ai & Bi) | ((Ai ^ Bi) & Cin)

Counting number of 1's in binary representation

- Let C(n) be the number of 1's in n
- What is the relation between C(n) and C(n/2)?
  - When n is even
  - When n is odd

Next: functions, arrays, String basics