15-213 S10 Recitation #1

Joel Feinstein, Carnegie Mellon University, Jan 2010
Parts adapted from 15-213S09Rec#1, Alex Gartrell, 2009
and 15213F08, “Bits, Bytes and Integers”, Greg Kesden, 2008
TA Intro
Recitation Policies

- Students may attend any recitation
- Recitation attendance is highly encouraged
- Often reveal important tips and tricks for labs
Fish Machines

- Computer cluster donated by Intel
  - You can’t walk into this cluster
- Login instructions on course web page under “Lab Machines”
  - [http://www.cs.cmu.edu/~213/labmachines.html](http://www.cs.cmu.edu/~213/labmachines.html)
- TEST YOUR FISH MACHINE ACCOUNT
- Send staff any issues
More Fish Machine

- Your labs will be graded on the fish machines
- If your lab does not work on the fish machines, you will not get credit
- You have been warned
- Check your grade on Autolab
Data Transmission to Fish Machines

- Share data storage with Andrew AFS
- Access using SFTP, SCP
- Or use your favorite FTP program
  - Fetch, Transmission on Mac
  - FileZilla, SSH Secure Shell on Windows
  - Have fun on Linux!
- See [http://www.cmu.edu/computing/software/all/index.html](http://www.cmu.edu/computing/software/all/index.html) for CMU-provided software
Autolab

- Serves as portal for:
  - Lab materials
  - Grading
  - Forums
  - Class status
  - Friendly competition!
  - (Class ranking is just for fun. Please do not hurt anybody.)

- http://autolab.cs.cmu.edu
TEST YOUR AUTOLAB ACCOUNT

If your account is not working, send an email to staff
  ◦ 15-213-staff@cs.cmu.edu

Datalab is available now!
Grading w/ Autolab

- Autolab will output a basic functionality score
  ◦ I.e. Does your code do what it is supposed to do?
- Style points are awarded based upon TA review
  ◦ How well do you document or comment your code?
  ◦ Is the organization thoughtful or is the code sloppy?
- If your code cheats the system, points may be removed
- Once you have full function points, any additional performance is for bragging rights
- Questions?
**GDB**

- Important tool when debugging programs
- Important commands
  - Break – Stop execution at line or function
  - Print – Display contents of variable
    - Whole slew of modifications: p/o for octal, p/c for char, etc.
  - X – Examine contents of memory
  - S – Step through program
- GDB can accept many commands with operations
  - i.e. +, -, &, *
GDB

- Compile with -g flag
  - gcc -g <FILENAME>
- Execute “gdb <EXECUTABLE>”
- Use run command to run program
GDB

[jnfeinst@unix33 ~]$ gcc -g test.c
[jnfeinst@unix33 ~]$ gdb ./a.out
GNU gdb Fedora (6.8-37.el5)
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu"...
(gdb) break main
Breakpoint 1 at 0x4004a0: file test.c, line 11.
(gdb) run
Starting program: /afs/andrew.cmu.edu/usr21/jnfeinst/test/a.out

Breakpoint 1, main () at test.c:11
11 char WHAT = 0x1;
(gdb) print WHAT
$1 = 0 '0'
(gdb) s
12 printf("\n\n\nWHAT = %d\n\n\n", WHAT);
(gdb) x &WHAT
0x7fff1f9eb9ef: 0x00000001
(gdb)
Binary-Hexadecimal

- Most common data formats at hardware level
- FYI Windows Vista/7 and Mac OS X (and Linux) have built in programmer calculators that can do these conversions

<table>
<thead>
<tr>
<th>Decimal: 15213</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary: 0011 1011 0110 1101</td>
</tr>
<tr>
<td>Hex: 3 B 6 D</td>
</tr>
</tbody>
</table>
How to convert DEC->BIN

- \(150_{10} = ?_2\)
- \(150_{10} = 2^7_{10} + 2^4_{10} + 2^2_{10} + 2^1_{10}\)
- \(= 10000000_2 + 00010000_2 + 00000100_2 + 00000010_2\)
- \(= 10010110_2\)
How to convert BIN->HEX

- $10010110_2 = ?_{16}$
- Group into 4 bits and convert to a hex character
- $1001_2 = 9_{10} = 9_{16}$
- $0110_2 = 6_{10} = 6_{16}$
- $10010110_2 = 96_{16}$
- Often denoted as 0x96
Signed and Unsigned Integers

- Both are a series of 32 bits
- Difference is in interpretation
- Unsigned
  - Every number is one more than the previous for whole range
  - Max = ? Min = ?
- Signed
  - Uses two’s complement interpretation
  - Half of the range is positive, half is negative
  - Max = ? Min = ?
Two’s Complement Subtraction

- Can perform subtraction without using subtraction!
- `char X = 150; char Y = 18;`
- `-Y = ~Y + 1 (two’s complement definition)`
- `X – Y = X + (~Y + 1)`
- `= 1001_0110 + (~0001_0010 + 1)`
- `= 1001_0110 + (1110_1101 + 1)`
- `= 1001_0110 + 1110_1110`
- `= 1_1000_0110 → 1000_0110 (char has 8 bits)`
- `= 132 (150 – 18 is in fact 132)`
Is char signed or unsigned?

- How would we find out?
Is char signed or unsigned?

```c
#include <stdlib.h>
#include <stdio.h>

int main() {
    char WHAT = 0xFF;
    printf("WHAT = %d\n", WHAT);
}
```

```
[jnfeinst@unix33 ~/test]$ ./a.out
WHAT = -1
```
Over-shifting

- What happens when you shift greater than the word width?

```c
int main() {
    int WHAT = 0x1;
    WHAT <<= 640;
    printf("\n\n\nWHAT = %d\n\n\n\n", WHAT);
}
```
Over-shifting

- Shift reduced modulo word width

```
[jnfeinst@unix33 ~/test]$ gcc test.c
 compiling...
 test.c: In function 'main':
 test.c:12: warning: left shift count >= width of type
 [jnfeinst@unix33 ~/test]$ ./a.out

WHAT = 1
```