Machine Arithmetic

8-31-2006

Opening Discussion

- Sorry I let you out early last class. My brain was stuck on an MWF schedule.
- What did we talk about last class?
- Do you have any questions about the reading?

Getting to Linux from Dorms

- One of the advantage of Linux is that it can be accessed remotely. You don't want to do that with these machines because they are dual-boot and people are likely to reboot them. However, we have a number of Linux only machines that you can safely log into from your dorm room and work on. (Xena01-Xena21)
- To connect to those machines you can use either Putty or a Linux terminal like Cygwin. There are links to both of these on the links page. Putty is simpler and just gives you a connection. Cygwin makes your Windows machine act like a Linux box.

Working with vi

- Now that we have moved around in Linux some, it is time to move to editing text files. I will ask you to use vi as your text editor in this class. It is a nice text editor for programming because it was built for that purpose.
- First we should use vi to edit some of the settings of vi. This will make vi more functional for coding.
- vi has two modes: command mode and edit mode (technically there are several types of edit mode). You start off in command mode. Typing in command mode doesn't actually type, it gives commands. Certain commands cause you to enter an edit mode. Press esc to get back to command mode.

Writing a C Program

- Now I want you all to go into your class directory and use vi to edit a file for a C program. You can call the file whatever you want, but it needs to end in ".c".
- Let's write a program that lets you input two integers and then prints out their sum, difference, product, and quotient.

Redirection Under Linux

- We have talked about some of the power of command line processing. Being able to add arguments to commands is a big benefit we have talked about.
- Another benefit is that we can redirect input and output in Linux. This means we can have programs write to files instead of to the screen. We can also have them read from files instead of the keyboard.
- We even have the ability to make the output of one program become the input of a second program. Let's look at this.

Math on Computers

- We've talked about the fact that computers use binary numbers for representing everything. Your book even went into some detail on the different types of numbers which we will hit on more next class.
- How would you do some of your standard math operations using binary numbers? Can you do addition in binary? What about multiplication?
- Let's look at these two operations first. Subtraction will wait until next class. How about division?
- What types of operations are really simple in binary?

Negative Integers

- Your book covers three ways that we could do negative numbers on a computer. What are they?
- Obviously the computer only does one. Which method to modern machines use and why?
- Let's go into more detail for this method and produce the negative binary forms of some different numbers.

Types in C

- The book goes into a fair bit of detail on types in C. This is a good thing and if you continue in CS you will need to know these details.
- For this class we will likely only use four types.
  - bool when we need just true or false
  - char when we need characters
  - int if we need integers
  - double if we need fractional values (floating points)
- We typically won't care about “wide” characters, different sized integers, unsigned integers, or single precision floating point values.
- I could put complex into some assignments if I can think of some good examples.

Floating Point Values

- Most modern machines have built in ability to handle fractional values with floating point calculations.
- Floating point can be contrasted to fixed point. The latter has a certain number of bits above and below a binary point (it's not a decimal point when you aren't working in decimal).
- Floating point is represented like scientific notation in binary. The main thing you should know about it is that these aren't truly real numbers. They aren't continuous and math with them is imprecise.

Testing Math Operations in C

- We can use C to test some of the things we have talked about. This is because C has ways to understand hexadecimal numbers and to print them out as well. We can also play tricks that I don't expect you to understand to print floating point values as hexidecimals.
- You book talks about printing integers by putting %d in format strings for printf. If you use %x you will get the value in hex. This and other things can be found in the man pages as well as in later chapters.

Expressions and Statements

- So far we've just been writing C code without really explaining much about it. Programming languages in general have a rigid syntax, they don't have all the exceptions of written languages.
- C, like most other programming languages, can be described as being built from statements, that are made from expressions that are built from tokens.
- Let's go look at some of the code we did today and see if we can pick out the pieces. Your reading for next class will present things more formally.

Minute Essay

- Add the following three binary numbers? Assume the answer is going into an 8 bit value so there are no issues with overflow.
  - 111
  - 110
  - 011
- Remember to read chapter 3 for Tuesday. Also keep in mind that the first quiz is on Tuesday and assignment #1 is due in a week.