CSC 180 lab 11: Function pointers

Thurs Nov 22 or Mon Nov 26, 2001

I don't have as much for you this week. I think a good thing to explore in the lab this week would be function pointers, as needed for assignment three. The lab TAs can assist you with the function pointer material in assignment three.

From talking with students I see that many have left out one lab or another due to time constraints. If you haven't done all the labs so far, I recommend spending some time this week on the earlier labs. All of them cover things you will want to know to be able to call yourself a C programmer.

Here are some function pointer exercises.

1. Write a function called ‘‘call’’ which takes two parameters, one a pointer to function with one int parameter and returning int, and the other an int. It calls the function and returns its value. This is a sufficiently trivial example to be quite silly, but it is probably a good first exercise.

So for example,

```
int xplusthree(int x)
{
    return x + 3;
}

int main()
{
    printf("%d\n", call(xplusthree, 58));
    return 0;
}
```

should print 61.

2. Write a function which sums the values of a function in a certain range. That is, you pass in a function pointer, and a "low" and a "high" value, and it calls the function high−low+1 times (low to high inclusive) and adds up the return values.

3. If you have time and have done the other labs to date, write a function which implements one of the root-finding algorithms to find a zero of an arbitrary function which is passed in. The function which is passed in will accept a double parameter and return a double value. The bisection method or secant method are reasonable algorithms to implement in this way with function pointers; for Newton's method, you'd need a second parameter which is a function which returns the derivative (at a given point; again, a double parameter and a double return value).