Tip 46: Typedefs save the day

Ben Klemens

1 January 2012

level: intermediate
purpose: write clearly

You may have noticed that I use this form for declaring structs every time:

```
typedef struct {
    int a, b;
    double c, d;
} newstruct_t;
```

This declares a new type that happens to be a structure of the given form.

The only reason to vary from this form is if you've got a struct that includes one of its own kind as an element (like how the `next` pointer of a linked list structure is to another linked list structure). In this case you need a slightly more elaborate form:

```
typedef struct _newstruct{
    int a, b;
    double c, d;
    struct _newstruct *next;
} newstruct_t;
```

Having a one-word name for the structure means less to worry about when you use the structure later, and in the same way, typedefs can clarify lots of other situations as well. Any time you find yourself putting together a complex type, which may just mean a pointer-to-pointer-to-pointer sort of situation, ask yourself whether a typedef could clarify things.

You already saw one example: Tip #17 (Entry #067) suggested defining

```
typedef char* string;
```

to reduce the visual clutter around arrays of strings and clarify their intent.

Defining function types can be especially confusing, even though the syntax is conceptually not that hard. If you have a function with a header like:

```
double a_fn(int, int); //a declaration
```

then just add a star and parens to describe its type:

```
double (*a_fn)(int, int)     //a type
```

But once you start putting this type into a declaration line, things start to get confusing again, so put `typedef` in front of that and forget about it:

```
typedef double (*a_fn_type)(int, int);
```

Now you can use it as a type like any other, such as to declare a function that takes another function as input:

```
double apply_a_fn(a_fn_type f, int first_in, int second_in){
    return f(first_in, second_in);
}
```

Suddenly writing functions that take other functions as inputs goes from a daunting test of star placement to kind of trivial. Pages 190*ff* of *Modeling with Data* present a full exposition.