Printing floating point numbers

IV. Postlude

The standard C library never seems to do quite what you want for printing floats. If you want scientific notation, you can use "%e", but then 0 prints as 0.000000e+00. Or you can use %f, but then large numbers yield long strings of digits rather than the scientific notation you'd prefer.

As a parting gift, here's a routine that prints real numbers a little more nicely, automatically adjusting format codes depending on what kind of number you give it. You can specify how big or small a number can get before moving to scientific notation, and you can still specify field widths as in the usual "%n.nf" format.

```
#include <ieee754.h>
#define LOG2_10 3.321928095

#define flt_zero(x) (fabs(x) < EPSILON)

int max_digs_rt = 3;  /* maximum # of 0's right of decimal before using
                         scientific notation */
int max_digs_lf = 5;  /* max # of digits left of decimal */

void print_real(double r, int width, int dec)
{
    int mag;
    double fpart, temp;
    char format[8];
    char num_format[3] = {'l',0,0};
    union ieee754_double *dl;

    dl = (union ieee754_double*)&r;
    mag = (dl->ieee.exponent - IEEE754_DOUBLE_BIAS) / LOG2_10;
    if (r == 0)
        mag = 0;
    if ((mag > max_digs_lf-1) || (mag < -max_digs_rt)) {
        num_format[1] = 'e';
        temp = r/pow(10, mag);      /* see if number will have a decimal */
        fpart = temp - floor(temp); /* when written in scientific notation */
    }
    else {
        num_format[1] = 'f';
        fpart = r - floor(r);
    }
    if (flt_zero(fpart))
        dec = 0;
    if (width == 0) {
        snprintf(format, 8, "%%.%d%s", dec, num_format);
    }
    else {
        snprintf(format, 8, "%%%d.%d%s", width, dec, num_format);
    }
    printf(format, r);
}
```