Hash Functions

A comprehensive collection of hash functions, a hash visualiser and some test results [see Mckenzie et al. *Selecting a Hashing Algorithm*, SP&E 20(2):209-224, Feb 1990] will be available someday. If you just want to have a good hash function, and cannot wait, `djb2` is one of the best string hash functions i know. it has excellent distribution and speed on many different sets of keys and table sizes. you are not likely to do better with one of the "well known" functions such as PJW, K&R[1], etc. Also see tpop pp. 126 for graphing hash functions.

---

djb2

this algorithm (k=33) was first reported by dan bernstein many years ago in comp.lang.c. another version of this algorithm (now favored by bernstein) uses xor: `hash(i) = hash(i - 1) * 33 ^ str[i];` the magic of number 33 (why it works better than many other constants, prime or not) has never been adequately explained.

```
unsigned long
hash(unsigned char *str)
{
    unsigned long hash = 5381;
    int c;

    while (c = *str++)
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */

    return hash;
}
```

---

sdbm

this algorithm was created for sdbm (a public-domain reimplementation of ndbm) database library. it was found to do well in scrambling bits, causing better distribution of the keys and fewer splits. it also happens to be a good general hashing function with good distribution. the actual function is `hash(i) = hash(i - 1) * 65599 + str[i];` what is included below is the faster version used in gawk. [there is even a faster, duff-device version] the magic constant 65599 was picked out of thin air while experimenting with different constants, and turns out to be a prime. this is one of the algorithms used in berkeley db (see sleepycat) and elsewhere.

```
static unsigned long
sdbm(str)
unsigned char *str;
{
    unsigned long hash = 0;
    int c;

    while (c = *str++)
        hash = c + (hash << 6) + (hash << 16) - hash;

    return hash;
}
```

---

lose lose

This hash function appeared in K&R (1st ed) but at least the reader was warned: "*This is not the best possible algorithm, but it has the merit of extreme simplicity*." This is an understatement; It is a *terrible* hashing algorithm, and it could have been much better without sacrificing its "extreme simplicity." [see the second edition!] Many C programmers use this function without actually testing it, or checking something like Knuth's *Sorting and Searching*, so it stuck. It is now found mixed with otherwise respectable code, eg. cnews. sigh. [see also: tpop]

```
unsigned long
hash(unsigned char *str)
{
    unsigned int hash = 0;
    int c;

    while (c = *str++)
        hash += c;

    return hash;
}
```