sdbm — Substitute DBM
or
Berkeley *ndbm* for Every UN*X[1] Made Simple

Ozan (oz) Yigit

The Guild of PD Software Toolmakers
Toronto - Canada

oz@nexus.yorku.ca

Implementation is the sincerest form of flattery. — L. Peter Deutsch

A The Clone of the *ndbm* library

The sources accompanying this notice — *sdbm* — constitute the first public release (Dec. 1990) of a complete clone of the Berkeley UN*X *ndbm* library. The *sdbm* library is meant to clone the proven functionality of *ndbm* as closely as possible, including a few improvements. It is practical, easy to understand, and compatible. The *sdbm* library is not derived from any licensed, proprietary or copyrighted software.

The *sdbm* implementation is based on a 1978 algorithm [Lar78] by P.-A. (Paul) Larson known as ‘‘Dynamic Hashing’’. In the course of searching for a substitute for *ndbm*, I prototyped three different external-hashing algorithms [Lar78, Fag79, Lit80] and ultimately chose Larson's algorithm as a basis of the *sdbm* implementation. The Bell Labs *dbm* (and therefore *ndbm*) is based on an algorithm invented by Ken Thompson, [Tho90, Tor87] and predates Larson's work.

The *sdbm* programming interface is totally compatible with *ndbm* and includes a slight improvement in database initialization. It is also expected to be binary-compatible under most UN*X versions that support the *ndbm* library.

The *sdbm* implementation shares the shortcomings of the *ndbm* library, as a side effect of various simplifications to the original Larson algorithm. It does produce *holes* in the page file as it writes pages past the end of file. (Larson's paper include a clever solution to this problem that is a result of using the hash value directly as a block address.) On the other hand, extensive tests seem to indicate that *sdbm* creates fewer holes in general, and the resulting pagefiles are smaller. The *sdbm* implementation is also faster than *ndbm* in database creation. Unlike the *ndbm*, the *sdbm* `store` operation will not ‘‘wander away’’ trying to split its data pages to insert a datum that *cannot* (due to elaborate worst-case situations) be inserted. (It will fail after a pre-defined number of attempts.)

Important Compatibility Warning

The *sdbm* and *ndbm* libraries *cannot* share databases: one cannot read the (dir/pag) database created by the other. This is due to the differences between the *ndbm* and *sdbm* algorithms[2], and the hash functions used. It is easy to convert between the *dbm/ndbm* databases and *sdbm* by ignoring the index completely: see `dbd`, `dbu` etc.

Notice of Intellectual Property

The entire sdbm *library package, as authored by me,* Ozan S. Yigit, *is hereby placed in the public domain.* As such, the author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from defects in it. There is no expressed or implied warranty for the *sdbm* library.

---

[1] UN*X is not a trademark of any (dis)organization.

[2] Torek's discussion [Tor87] indicates that *dbm/ndbm* implementations use the hash value to traverse the radix trie differently than *sdbm* and as a result, the page indexes are generated in *different* order. For more information, send e-mail to the author.

Since the *sdbm* library package is in the public domain, this *original* release or any additional public-domain releases of the modified original cannot possibly (by definition) be withheld from you. Also by definition, You (singular) have all the rights to this code (including the right to sell without permission, the right to hoard[3] and the right to do other icky things as you see fit) but those rights are also granted to everyone else.

Please note that all previous distributions of this software contained a copyright (which is now dropped) to protect its origins and its current public domain status against any possible claims and/or challenges.

Acknowledgments

Many people have been very helpful and supportive. A partial list would necessarily include Rayan Zacherissen (who contributed the man page, and also hacked a MMAP version of *sdbm*), Arnold Robbins, Chris Lewis, Bill Davidsen, Henry Spencer, Geoff Collyer, Rich Salz (who got me started in the first place), Johannes Ruschein (who did the minix port) and David Tilbrook. I thank you all.

Distribution Manifest and Notes

This distribution of *sdbm* includes (at least) the following:

```
CHANGES     change log
README      this file.
biblio      a small bibliography on external hashing
dba.c       a crude (n/s)dbm page file analyzer
dbd.c       a crude (n/s)dbm page file dumper (for conversion)
dbe.1       man page for dbe.c
dbe.c       Janick's database editor
dbm.c       a dbm library emulation wrapper for ndbm/sdbm
dbm.h       header file for the above
dbu.c       a crude db management utility
hash.c      hashing function
makefile    guess.
pair.c      page-level routines (posted earlier)
pair.h      header file for the above
readme.ms   troff source for the README file
sdbm.3      man page
sdbm.c      the real thing
sdbm.h      header file for the above
tune.h      place for tuning & portability thingies
util.c      miscellaneous
```

`dbu` is a simple database manipulation program[4] that tries to look like Bell Labs' `cbt` utility. It is currently incomplete in functionality. I use `dbu` to test out the routines: it takes (from stdin) tab separated key/value pairs for commands like `build` or `insert` or takes keys for commands like `delete` or `look`.

```
dbu <build|creat|look|insert|cat|delete> dbmfile
```

`dba` is a crude analyzer of *dbm/sdbm/ndbm* page files. It scans the entire page file, reporting page level statistics, and totals at the end.

`dbd` is a crude dump program for *dbm/ndbm/sdbm* databases. It ignores the bitmap, and dumps the data pages in sequence. It can be used to create input for the `dbu` utility. Note that `dbd` will skip any

---

[3] You cannot really hoard something that is available to the public at large, but try if it makes you feel any better.

[4] The `dbd`, `dba`, `dbu` utilities are quick hacks and are not fit for production use. They were developed late one night, just to test out *sdbm*, and convert some databases.

NULLs in the key and data fields, thus is unsuitable to convert some peculiar databases that insist in including the terminating null.

I have also included a copy of the `dbe` (*ndbm* DataBase Editor) by Janick Bergeron [janick@bnr.ca] for your pleasure. You may find it more useful than the little `dbu` utility.

`dbm.[ch]` is a *dbm* library emulation on top of *ndbm* (and hence suitable for *sdbm*). Written by Robert Elz.

The *sdbm* library has been around in beta test for quite a long time, and from whatever little feedback I received (maybe no news is good news), I believe it has been functioning without any significant problems. I would, of course, appreciate all fixes and/or improvements. Portability enhancements would especially be useful.

Implementation Issues

Hash functions: The algorithm behind *sdbm* implementation needs a good bit-scrambling hash function to be effective. I ran into a set of constants for a simple hash function that seem to help *sdbm* perform better than *ndbm* for various inputs:

```
/*
 * polynomial conversion ignoring overflows
 * 65599 nice. 65587 even better.
 */
long
dbm_hash(char *str, int len) {
    register unsigned long n = 0;

    while (len--)
        n = n * 65599 + *str++;
    return n;
}
```

There may be better hash functions for the purposes of dynamic hashing. Try your favorite, and check the pagefile. If it contains too many pages with too many holes, (in relation to this one for example) or if *sdbm* simply stops working (fails after `SPLTMAX` attempts to split) when you feed your NEWS `history` file to it, you probably do not have a good hashing function. If you do better (for different types of input), I would like to know about the function you use.

Block sizes: It seems (from various tests on a few machines) that a page file block size `PBLKSIZ` of 1024 is by far the best for performance, but this also happens to limit the size of a key/value pair. Depending on your needs, you may wish to increase the page size, and also adjust `PAIRMAX` (the maximum size of a key/value pair allowed: should always be at least three words smaller than `PBLKSIZ`.) accordingly. The system-wide version of the library should probably be configured with 1024 (distribution default), as this appears to be sufficient for most common uses of *sdbm*.

Portability

This package has been tested in many different UN*Xes even including minix, and appears to be reasonably portable. This does not mean it will port easily to non-UN*X systems.

Notes and Miscellaneous

The *sdbm* is not a very complicated package, at least not after you familiarize yourself with the literature on external hashing. There are other interesting algorithms in existence that ensure (approximately) single-read access to a data value associated with any key. These are directory-less schemes such as *linear hashing* [Lit80] (+ Larson variations), *spiral storage* [Mar79] or directory schemes such as *extensible hashing* [Fag79] by Fagin et al. I do hope these sources provide a reasonable playground for experimentation with other algorithms. See the June 1988 issue of ACM Computing Surveys [Enb88] for an excellent overview of the field.

References

[Lar78] P.-A. Larson, ‘‘Dynamic Hashing’’, *BIT*, vol. 18, pp. 184-201, 1978.

[Tho90] Ken Thompson, *private communication*, Nov. 1990

[Lit80] W. Litwin, ‘‘ Linear Hashing: A new tool for file and table addressing’’, *Proceedings of the 6th Conference on Very Large Dabatases (Montreal)*, pp. 212-223, Very Large Database Foundation, Saratoga, Calif., 1980.

[Fag79] R. Fagin, J. Nievergelt, N. Pippinger, and H. R. Strong, ‘‘Extendible Hashing - A Fast Access Method for Dynamic Files’’, *ACM Trans. Database Syst.*, vol. 4, no.3, pp. 315-344, Sept. 1979.

[Wal84] Rich Wales, ‘‘Discussion of "dbm" data base system’’, *USENET newsgroup unix.wizards*, Jan. 1984.

[Tor87] Chris Torek, ‘‘Re: dbm.a and ndbm.a archives’’, *USENET newsgroup comp.unix*, 1987.

[Mar79] G. N. Martin, ‘‘Spiral Storage: Incrementally Augmentable Hash Addressed Storage’’, *Technical Report #27*, University of Varwick, Coventry, U.K., 1979.

[Enb88] R. J. Enbody and H. C. Du, ‘‘Dynamic Hashing Schemes’’,*ACM Computing Surveys*, vol. 20, no. 2, pp. 85-113, June 1988.