

FAQ's

If you can't find the answers to your questions on this page or the website as a whole please contact us by email, fax or telephone. Our contact information is available via the *Contact Us* link at the bottom of the page.

- [Multi-threading deadlock problems](#)
- [Is D-ISAM compatible with Informix C-ISAM?](#)
- [Performance Comparisons](#)
- [I'm sure I have compiled the library properly, and I'm sure I have compiled and linked my application properly but nothing seems to work correctly.](#)
- [I'm sure I have compiled the library properly, and I am sure I have the right linker commands, but I keep getting unresolved references to the base \(isXxxx\(\) \) library functions.](#)
- [What's involved in converting from C-ISAM or D-ISAM 3.x to DISAM96/DISAM7?](#)
- [What are the maximums allowable with DISAM?](#)
- [What are the differences between DISAM 3.x and DISAM96 and DISAM7?](#)
- [Rebuilding and reclaiming space using Disam's dpack](#)
- [Size of resulting index versus size of data](#)
- [Can DISAM96/DISAM7 be built as a 64-bit library to go into a 64-bit application?](#)

This problem may be occurring in other Linux type distributions as well.

If you're running a multi-threaded DISAM program on Linux Ubuntu, with multiple processors and are seeing what appears to be a deadlock scenario, try switching to the latest Ubuntu 9.1. We have been able to confirm that some kernel versions that run NPTL threads on multi-processor systems run into what appears to be a bug in the kernel or NPTL threads library, specifically a `futex_lock` failure that kills the thread leaving other waiting threads in a permanent wait condition. To date we have not been able to duplicate this bug with the latest version of Ubuntu 9.1, specifically kernel 2.6.31-14 server. NPTL version 2.10.1.

Is D-ISAM compatible with Informix C-ISAM?

D-ISAM 3.x, DISAM96 and DISAM7 are all file compatible with C-ISAM. D-ISAM 3.x, DISAM96/DISAM7 will run concurrent locking with C-ISAM Version 4.1 executables. DISAM96 and DISAM7 allows for configuration settings that will enable concurrent locking with C-ISAM 5.1 and greater. (Currently this applies to unix fcntl() environments only) This is only of importance if you are planning to run C-ISAM executables against D-ISAM/DISAM96/DISAM7 executables. The D-ISAM 3.x calls and DISAM96/DISAM7 wrap calls are fully compatible with the standard C-ISAM isam function calls.

Performance Comparisons

Some Performance Comparisons between D-ISAM Version 3.x, DISAM96 and Informix' C-ISAM 4.x
 Tests were 15000 cycles run on an Intel Pentium-133 running SCO Unix ODT-5

**Some Performance Comparisons
 between D-ISAM Version 3.x, DISAM96 and Informix' C-ISAM 4.x**
 Tests were 15000 cycles run on an Intel Pentium-133 running SCO
 Unix ODT-5

	D-ISAM 3.x	DISAM96	C-ISAM 4.x
Write cycle	22.00	13.00	13.66
Read cycle	13.66	9.33	9.33
Update cycle	39.00	18.33	16.33
Delete cycle	36.33	22.66	22.33

I'm sure I have compiled the library properly, and I'm sure I have

compiled and linked my application properly but nothing seems to work correctly.

A. Ensure your including the correct header information, if compiling or converting from C-ISAM you'll need to change references from isam.h to disam.h (versions 3.x) or iswrap.h (DISAM96/DISAM7)...or simply create a dummy isam.h that only includes #include or .

B. Disam requires that that library be compiled with signed char types as the default. Ensure the compiler is defaulting to signed char types on compile or set the compiler flag that generates signed char types by default.

I'm sure I have compiled the library properly, and I am sure I have the right linker commands, but I keep getting unresolved references to the base (isXxxx()) library functions.

DISAM96/DISAM7 specific – If you're linking wrap (isxxxx()) calls then it is possible your linker resolves references canonically – in a single pass through the listed libraries. Linker objects should be listed in order of precedence – object module(s), wrap lib, base lib.

What's involved in converting from C-ISAM or D-ISAM 3.x to DISAM96/DISAM7?

If converting from C-ISAM to D-ISAM 3.x the only change you will need to make is to create an isam.h that only includes . Compile the D-ISAM 3.x library and recompile your application code and link to the D-ISAM library.

If converting from C-ISAM or D-ISAM 3.x to DISAM96 or DISAM7, you will need to ensure that references to isam.h or disam.h are replaced with iswrap.h. DISAM96 and DISAM7 no longer use the isfdmap structure that is internal to D-ISAM 3.x, DISAM96 and DISAM7 provide analogs for some of the more common values via common dictionary access routines.

What are the maximums allowable with DISAM?

FAQ's

Most maximums such as file sizes/open files and such are determined by the os. DISAM96/DISAM7 will allow index sizes limited only by the index block factor, to increase the default maximum for the index size increase the ISIDXBLK default setting by increments of 512 bytes..the default is set to 1024 and allows a max index size of around 990 bytes. DISAM96/DISAM7 as of version 6.12 allows support for files greater than the standard 2 gig limit for those operating systems that support int64 types.

DISAM sets no limits for the number of indexes in a file or the number of parts per key.

What are the differences between DISAM 3.x and DISAM96 and DISAM7?

DISAM96 is a complete rewrite that resembles D-ISAM 3.x in almost no way as far as the actual code is concerned. DISAM96 is cleaner, more efficient code with many added options such as multithread capabilities, large file support for platforms that allow > 2 gig file sizes, locking compliance with both C-ISAM 4.1 as well as C-ISAM 5.0 and 7.1. (C-ISAM changed locking schemes as of 5.0), full cobol isstat variable support, MF cobol filename support, and ease in compilation of both static and dll libraries for MS platforms.

DISAM7 is the latest DISAM next generation of DISAM96. DISAM7 allows twice the number of data records compared with D-ISAM and DISAM96. DISAM7 allows for isam file schema storage within the isam file, formatted reported output of data from the file via the schema as well as an enhancement to the C-ISAM limit on variable length record max, see dvlreblid.ref for more details.

Rebuilding and reclaiming space using Disam's dpack

When a record is deleted from an isam file, it is marked for deletion in the .dat and the corresponding index values are physically removed from the index node. (One 512 or 1024 block = one index record). When an index node becomes empty due to all key values in the node being deleted, then the node itself is removed from the .idx file.

At any time an index node may or may not be full, if a deletion leaves space

FAQ's

in an index node..that space will only be reused if a new record insert has a key value that would compare as next or prev against existing key values in that node. In other words an index node (used bytes) shrinks or expands(up to 512 or 1024 max bytes) due to deletions and insertions. An insertion does not reuse any available index node with available space..it must be the index node where the comparable key values reside.

Depending on the index values on new record inserts, index node space that has been freed from previously deleted entries may not ever be reused unless new key values compare as the next key value to an existing key value in that node.

There is no ratio or percentage that can be assumed about how much unused space is in a index node. That depends on the entries deleted and subsequent inserts if any. If all records in a file are removed the .idx file will shrink in size because nodes will be empty and empty nodes are physically deleted.

The only way to pack both the .dat and .idx is to run Disam96's dpack program.

Size of resulting index versus size of data

512 index node size, dup width of 2 bytes and assuming no index compression: 100 byte key value, with 2 keys

one 512 byte node would be allocated for every 4 – 100 byte index values (there is extra space per node 2 bytes for the #bytes used in node as well as 4 bytes per value for record# and 2 bytes per value for dup#) so 10 records of 100 bytes would mean 4 index values per node = 3 nodes * 512bytes * # file indexes

key length – 100 bytes + 4 bytes – record# + 2 bytes – dup# -----
106 bytes per index value

512 – index header record + 3 * 512 – 10 index values for key 1 + 3 * 512 – 10 index values for key 2 ----- 3584 bytes for index values + header record for .idx file

and 101 per data record * 10 records = 1010 bytes for the .dat file.

1024 index block size and 4 byte dup width (now standard) 9 index values in 1 node, another node for the 10th value

1024 - index header record + + 2 * 1024 - 10 index values for key 1 + 2 * 1024 - 10 index values for key 2 ----- 5120 bytes bytes for .idx

Once a node is full, that node is split into to 2 nodes and another level node is created. This forms the index chain. The first node is the level 0 node and level 0 nodes point directly to data records. Higher level nodes point to the index record of the lower level node in the index chain.

Can DISAM96/DISAM7 be built as a 64-bit library to go into a 64-bit application?

DISAM96 and DISAM7 can be compiled to allow file offsets which are normally defined as 32bit longs to be defined as int64 types..see definition of OFFT type in isport.h or isbase.h. By setting ISHUGE to 1 OFFT will be defined as int64_t. Check your compiler options for - D_LARGEFILE64_SOURCE and -D_FILE_OFFSET_BITS=64 or similar settings which would be required for compiling for 64bit in a 32bit environment. Do not set ISHUGE for native 64bit environments.