NAME

db_jump_set, db_value_set – replace underlying DB functionality

SYNOPSIS

#include <db.h>

int
db_jump_set(void *func, int which);

int
db_value_set(int value, int which);

DESCRIPTION

The DB library is a family of groups of functions that provides a modular programming interface to transactions and record-oriented file access. The library includes support for transactions, locking, logging and file page caching, as well as various indexed access methods. Many of the functional groups (e.g., the file page caching functions) are useful independent of the other DB functions, although some functional groups are explicitly based on other functional groups (e.g., transactions and logging). For a general description of the DB package, see *db_intro*(3).

db_jump_set

The *db_jump_set* function enables applications to replace underlying DB library functionality by replacing entries in a function call jump table. The *which* argument specifies the entry to be replaced by the argument *func*.

The following values of *which* are supported:

DB_FUNC_CLOSE

Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *close* function with *func*, which must conform to the standard interface.

DB_FUNC_DIRFREE

The DB library requires the ability to return any memory allocated as part of the routine which reads through a directory and creates a list of files that that the directory contains (see DB_FUNC_DIRLIST). The *func* argument must conform to the following interface:

```
int dirfree(char **namesp, int cnt);
```

The *namesp* and *cnt* arguments are the same values as were returned by the DB_FUNC_DIRLIST function.

The *dirfree* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_DIRLIST

The DB library requires the ability to read through a directory and create a list of files that that the directory contains. The *func* argument must conform to the following interface:

```
int dirlist(const char *dir,
        char ***namesp, int *cntp);
```

The *dir* argument is the name of the directory to be searched. The function must return a pointer to an array of nul-terminated file names in the memory location referenced by the argument *namesp*, and a count of the number of elements in the array in the memory location referenced by *cntp*.

The *dirlist* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_EXISTS

The DB library requires the ability to determine if a file exists, and optionally, if it is a file of type directory. The *func* argument must conform to the following interface:

```
int exists(const char *path, int *isdirp);
```

The *path* argument is the pathname of the file to be checked.

If the *isdirp* argument is non-NULL, it must be set to non-0 if *path* is a directory, and 0 if *path* is not a directory.

The *exists* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_FREE
Replace all DB calls to the ANSI C X3.159-1989 (“ANSI C”) standard *free* function with *func*, which must conform to the standard interface.

DB_FUNC_FSYNC
Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *fsync* function with *func*, which must conform to the standard interface.

DB_FUNC_IOINFO
The DB library requires the ability to determine the size and I/O characteristics of a file. The *func* argument must conform to the following interface:

```
int ioinfo(const char *path, int fd,
        u_int32_t *mbytesp, u_int32_t *bytesp,
        u_int32_t *iosizep);
```

The *path* argument is the pathname of the file to be checked, and the *fd* argument is an open file descriptor on the file.

If the *mbytesp* and *bytesp* arguments are non-NULL, the *ioinfo* function must return in them the size of the file: the number of megabytes in the file into the memory location referenced by the *mbytesp* argument, and the number of bytes over and above that number of megabytes into the memory location referenced by the *bytesp* argument.

In addition, if the *iosizep* argument is non-NULL, the *ioinfo* function must return the optimum granularity for I/O operations to the file in the memory location referenced by it.

The *ioinfo* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_MALLOC
Replace all DB calls to the ANSI C X3.159-1989 (“ANSI C”) standard *malloc* function with *func*, which must conform to the standard interface.

DB_FUNC_MAP
The DB library requires the ability to map a file into memory. The *func* argument must conform to the following interface:

```
int map(int fd, size_t len,
        int is_private, int rdonly, void **addr);
```

The *fd* argument is an open file descriptor on the file. The *map* function must map the first *len* bytes of the file into memory and return a pointer to the mapped location in the the memory location referenced by the argument *addr*.

The *is_private* argument will be non-zero if the file is private to the application, otherwise, the file must be mapped into shared memory accessible to any number of processes.

The *is_rdonly* argument will be non-zero if the file is considered read-only by the caller.

Both private and public files may contain semaphores.

The *map* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_OPEN
Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *open* function with *func*, which must conform to the standard interface.

DB_FUNC_READ
Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *read* function with *func*, which must conform to the standard interface.

DB_FUNC_REALLOC
Replace all DB calls to the ANSI C X3.159-1989 (“ANSI C”) standard *realloc* function with *func*, which must conform to the standard interface.

DB_FUNC_SEEK
The DB library requires the ability to specify that a subsequent read or write to a file will occur from a specific location in that file. The *func* argument must conform to the following interface:

```
int seek(int fd, size_t pgsize,
        db_pgno_t pageno, u_long relative, int whence);
```

The *fd* argument is an open file descriptor on the file. The *seek* function must cause a subsequent read or write from the file to occur at a byte offset specified by the calculation:

```
(pgsize * pageno) + relative
```

The *whence* argument specifies where in the file the byte offset is relative to, as described by the Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *function with func*, which must conform to the standard interface.
for the *lseek* system call.

The *seek* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_SLEEP
The DB library requires the ability to cause a process to suspend itself for a period of time, relinquishing control of the processor to any other waiting thread of control. The *func* argument must conform to the following interface:

```
int sleep(u_long seconds, u_long microseconds);
```

The *seconds* and *microseconds* arguments specify the amount of time to wait until the suspending thread of control should run again.

The *seconds* and *microseconds* arguments may not be normalized when the *sleep* function is called, i.e., the *microseconds* argument may be greater than 1000000.

The *sleep* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_STRDUP
Replace all DB calls to the ANSI C X3.159-1989 (“ANSI C”) standard *strdup* function with *func*, which must conform to the standard interface.

DB_FUNC_UNLINK
Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *unlink* function with *func*, which must conform to the standard interface.

DB_FUNC_UNMAP
The DB library requires the ability to unmap a file from memory. The *func* argument must conform to the following interface:

```
int unmap(void *addr, size_t len);
```

The *addr* argument is the argument returned by the DB_FUNC_MAP function when the file was mapped into memory, and the *len* argument is the same as the *len* argument specified to the DB_FUNC_MAP function when the file was mapped into memory.

The *unmap* function returns the value of *errno* on failure and 0 on success.

DB_FUNC_WRITE
Replace all DB calls to the IEEE Std 1003.1b-1993 (“POSIX”) standard *write* function with *func*, which must conform to the standard interface.

DB_FUNC_YIELD
The DB library requires the ability to yield the processor from the current thread of control to any other waiting threads of control. The *func* argument must conform to the following interface:

```
int yield(void);
```

The *yield* function must be able to cause the rescheduling all participants in the current DB environment, whether threaded or not. It may be incorrect to supply a thread *yield* function if more than a single process is operating in the DB environment. This is because many thread-yield functions will not allow other processes to run, and the contested lock may be held by another process, not by another thread.

If no *yield* function is specified, or if the *yield* function returns an error, the function specified by the DB_FUNC_SLEEP entry will be used instead or subsequently, i.e., if no *yield* function is specified, or it is possible for the *yield* function to fail, the *sleep* function **must** cause the processor to reschedule any waiting threads of control for execution.

Solaris architecture note: Because of bugs in versions of Solaris before version 5.6, the DB library uses the *sema_wait*(3T) call instead of the *sema_trywait*(3T) call. For this reason, replacing the *yield* function will have no effect on Solaris.

The *yield* function returns the value of *errno* on failure and 0 on success.

The *db_jump_set* function returns the value of *errno* on failure and 0 on success.

db_value_set
The *db_value_set* function enables applications to specify underlying DB library functionality. The *which* argument specifies the information being set by the argument *value*.

The following values of *which* are supported:

DB_TSL_SPINS
Specify the number of times mutexes should spin without blocking.

This value normally defaults to 1 (correct for uniprocessor systems), but on multiprocessor Win32 systems defaults to 50 times the number of processors. Otherwise, the number of spins should be set to an appropriate value when applications are running on multiprocessor systems.

The *db_value_set* function returns the value of *errno* on failure and 0 on success.

ERRORS

The *db_jump_set* function may fail and return *errno* for the following conditions:

[EINVAL]
An invalid flag value or parameter was specified.

BUGS

No type checking is done of the *func* argument, and specifying an invalid replacement routine will cause unpredictable results.

SEE ALSO

db_archive(1), *db_checkpoint*(1), *db_deadlock*(1), *db_dump*(1), *db_load*(1), *db_recover*(1), *db_stat*(1), *db_intro*(3), *db_appinit*(3), *db_cursor*(3), *db_dbm*(3), *db_internal*(3), *db_lock*(3), *db_log*(3), *db_mpool*(3), *db_open*(3), *db_thread*(3), *db_txn*(3)