

NAME

alq, alq_open, alq_write, alq_flush, alq_close, alq_get, alq_post — Asynchronous Logging Queues

SYNOPSIS

```
#include <sys/alq.h>

int
alq_open(struct alq **app, const char *file, int size, int count);

int
alq_write(struct alq *alq, void *data, int waitok);

void
alq_flush(struct alq *alq);

void
alq_close(struct alq *alq);

struct ale *
alq_get(struct alq *alq, int waitok);

void
alq_post(struct alq *alq, struct ale *ale);
```

DESCRIPTION

The **alq** facility provides an asynchronous fixed length recording mechanism, known as Asynchronous Logging Queues. It can record to any vnode(9), thus providing the ability to journal logs to character devices as well as regular files. All functions accept a *struct alq* argument, which is an opaque type that maintains state information for an Asynchronous Logging Queue. The logging facility runs in a separate kernel thread, which services all log entry requests.

An “asynchronous log entry”, is defined as *struct ale*, which has the following members:

```
struct ale {
    struct ale    *ae_next;    /* Next Entry */
    char          *ae_data;    /* Entry buffer */
    int           ae_flags;    /* Entry flags */
};
```

The *ae_flags* field is for internal use, clients of the **alq** interface should not modify this field. Behaviour is undefined if this field is modified.

FUNCTIONS

The **alq_open()** function creates a new logging queue.

The *file* argument is the name of the file to open for logging, The size of each entry in the queue is determined by *size*. The *count* argument determines the number of items to be stored in the asynchronous queue over an approximate period of a disk write operation.

The **alq_write()** function writes *data* to the designated queue, *alq*. In the event that **alq_write()** could not write the entry immediately, and ALQ_WAITOK is passed to *waitok*, then **alq_write()** will be allowed to `tsleep(9)`.

The **alq_flush()** function is used for flushing *alq* to the log medium that was passed to **alq_open()**.

The **alq_close()** function will close the asynchronous logging queue, *alq*, and flush all pending write requests to the log medium. It will free all resources that were previously allocated.

The `alq_get()` function returns the next available asynchronous logging entry from the queue, `alq`. This function leaves the queue in a locked state, until a subsequent `alq_post()` call is made. In the event that `alq_get()` could not retrieve an entry immediately, it will `tsleep(2)` with the “alqget” wait message.

The `alq_post()` function schedules the asynchronous logging entry, `ale`, which is retrieved using the `alq_get()` function, for writing to the asynchronous logging queue, `alq`. This function leaves the queue, `alq`, in an unlocked state.

IMPLEMENTATION NOTES

The `alq_open()` function uses the credentials of the invoking thread for opening files. The `alq_write()` function is a wrapper around the `alq_get()` and `alq_post()` functions; by using these functions separately, a call to `bcopy()` can be avoided for performance critical code paths.

RETURN VALUES

The `alq_open()` function returns one of the error codes listed in `open(2)`, if it fails to open `file`, or else it returns 0.

The `alq_write()` function returns `EWOULDBLOCK` if `ALQ_NOWAIT` was provided as a value to `waitok` and either the queue is full, or when the system is shutting down.

The `alq_get()` function returns `NULL`, if `ALQ_NOWAIT` was provided as a value to `waitok` and either the queue is full, or when the system is shutting down.

NOTE, invalid arguments to non-void functions will result in undefined behaviour.

SEE ALSO

`syslog(3)`, `kthread(9)`, `ktr(9)`, `tsleep(9)`, `vnode(9)`

HISTORY

The Asynchronous Logging Queues (ALQ) facility was first introduced in FreeBSD 5.0

AUTHORS

The `alq` facility was written by Jeffrey Roberson <jeff@FreeBSD.org>.

This manual page was written by Hiten M. Pandya <hmp@FreeBSD.org>.