NTNX HASH DEVICE LIBRARY

CONTENTS:

1)Function Definitions in the ntnx_hash.h header file

The following document describes the functions defined by the ntnx_hash library

ntnx_hash_setup

```
ntnx_hash_t *ntnx_hash_setup(void);
```

This function sets up the context for the ntnx hash functions by allocating memory for `ntnx_hash_t`, opening the device file `/dev/ntnx_hash`, and checking if the IOCTL API version is compatible.

Returns a pointer to the `ntnx_hash_t` context on success, or `NULL` on failure.

ntnx_hash_compute

```
char *ntnx_hash_compute(ntnx_hash_t *ctx, void *buf, size_t len);
```

This function computes the ntnx hash value of a given buffer using the `ntnx_hash_t` context. It allocates memory for a hash value string and sets up a `ntnx_hash_compute` struct with the buffer and length, then calls the IOCTL to compute the hash value.

Returns the hash value string on success, or `NULL` on failure.

ntnx_hash_destroy

```
int ntnx_hash_destroy(ntnx_hash_t *ctx);
```

This function destroys the `ntnx_hash_t` context by closing the device file and freeing the memory.

Returns `0` on success, or `-1` on failure.

2) Function Descriptions in ntnx_hash.c

The code defines the following constants at the beginning:

- NTNX_HASH_GET_API_VERSION: The value used to request the IOCTL API version.
- NTNX_HASH_DEVICE: The device file used for the ntnx hash functions.
- NTNX_HASH_COMPUTE: The value used to call the IOCTL to compute the hash value.

ntnx_hash_setup()

This function sets up the context for the ntnx hash functions by:

1. Allocating memory for the `ntnx_hash_t` context.
2. Opening the device file `/dev/ntnx_hash`.
3. Checking if the IOCTL API version is compatible.
4. Returning a pointer to the `ntnx_hash_t` context on success, or `NULL` on failure.

ntnx_hash_compute()

The function `ntnx_hash_compute` computes the hash value of a given buffer using a custom device driver. The function takes in a pointer to an `ntnx_hash_t` context which contains the file descriptor of the device driver, a pointer to the buffer and its length. It returns a pointer to a null-terminated string of 33 characters that represents the hash value.

The function first checks if the `ntnx_hash_t` context is valid. If it is not, it sets the error code to `EINVAL` and returns `NULL`.

Next, the function allocates memory for the hash value string. If the allocation fails, it sets the error code to `ENOMEM` and returns `NULL`.

The function then sets up a struct `ntnx_hash_compute` with the buffer pointer, buffer length, and the pointer to the allocated memory for the hash value string.

Finally, the function calls the `ioctl` system call with the device file descriptor, `NTNX_HASH_COMPUTE` command, and the address of the `ntnx_hash_compute` struct to compute the hash value. If the `ioctl` call fails, the function sets the error code to `ENOMEM` and returns `NULL`. Otherwise, it returns the pointer to the hash value string.

ntnx_hash_destroy()

The `ntnx_hash_destroy` function takes a pointer to an `ntnx_hash_t` context as an argument. It first checks if the context pointer is valid, and if not, it sets the `errno` variable to `EINVAL` and returns -1 to indicate an error.

If the context pointer is valid, it closes the device file associated with the context using the `close()` system call and frees the memory allocated for the context using the `free()` function. It then returns the value returned by the `close()` system call, which should be 0 on success and -1 on failure.