NAME

buf – block I/O data transfer structure

SYNOPSIS

#include <sys/types.h>
#include <sys/buf.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

DESCRIPTION

The **buf** structure is the basic data structure for block I/O transfers. Each block I/O transfer has an associated buffer header. The header contains all the buffer control and status information. For drivers, the buffer header pointer is the sole argument to a block driver **strategy**(9E) routine. Do not depend on the size of the **buf** structure when writing a driver.

It is important to note that a buffer header may be linked in multiple lists simultaneously. Because of this, most of the members in the buffer header cannot be changed by the driver, even when the buffer header is in one of the drivers' work lists.

Buffer headers are also used by the system for unbuffered or physical I/O for block drivers. In this case, the buffer describes a portion of user data space that is locked into memory.

Block drivers often chain block requests so that overall throughput for the device is maximized. The **av_forw** and the **av_back** members of the **buf** structure can serve as link pointers for chaining block requests.

STRUCTURE MEMBERS

```
int             b_flags;         /* Buffer status */
struct buf      *av_forw;        /* Driver work list link */
struct buf      *av_back;        /* Driver work list link */
unsigned int    b_bcount;        /* # of bytes to transfer */
union {
        caddr_t b_addr;          /* Buffer's virtual address */
} b_un;
daddr_t         b_blkno;         /* Block number on device */
diskaddr_t      b_lblkno;        /* Expanded block number on device */
unsigned int    b_resid;         /* # of bytes not transferred */
long            b_bufsize;       /* size of allocated buffer */
int             (*b_iodone)();   /* function called by biodone */
int             b_error;         /* expanded error field */
void            *b_private;      /* "opaque" driver private area */
dev_t           b_edev;          /* expanded dev field */
```

The members of the buffer header available to test or set by a driver are as follows:

b_flags stores the buffer status and tells the driver whether to read or write to the device. The driver must never clear the **b_flags** member. If this is done, unpredictable results can occur including loss of disk sanity and the possible failure of other kernel processes.

Valid flags are as follows:

B_BUSY indicates the buffer is in use. The driver may not change this flag unless it allocated the buffer with **getrbuf**(9F)**,** and no I/O operation is in progress.

B_DONE indicates the data transfer has completed. This flag is read-only.

B_ERROR indicates an I/O transfer error. It is set in conjunction with the **b_error** field. **bioerror**(9F) should be used in preference to setting the **B_ERROR** bit.

B_PAGEIO indicates the buffer is being used in a paged I/O request. See the description of the **b_un.b_addr** field for more information. This flag is read-only.

B_PHYS indicates the buffer header is being used for physical (direct) I/O to a user data area. See the description of the **b_un.b_addr** field for more information. This flag is read-only.

B_READ indicates data is to be read from the peripheral device into main memory.

B_WRITE indicates the data is to be transferred from main memory to the peripheral device. **B_WRITE** is a pseudo flag that occupies the same bit location as **B_READ**. **B_WRITE** cannot be directly tested; it is only detected as the NOT form of **B_READ**.

av_forw and **av_back** can be used by the driver to link the buffer into driver work lists.

b_bcount specifies the number of bytes to be transferred in both a paged and a non-paged I/O request.

b_un.b_addr is the virtual address of the I/O request, unless **B_PAGEIO** is set. The address is a kernel virtual address, unless **B_PHYS** is set, in which case it is a user virtual address. If **B_PAGEIO** is set, **b_un.b_addr** contains kernel private data. Note that either one of **B_PHYS** and **B_PAGEIO**, or neither, may be set, but not both.

b_blkno identifies which logical block on the device (the device is defined by the device number) is to be accessed. The driver may have to convert this logical block number to a physical location such as a cylinder, track, and sector of a disk. This is a 32-bit value. The driver should use **b_blkno** or **b_lblkno**, but not both.

b_lblkno identifies which logical block on the device (the device is defined by the device number) is to be accessed. The driver may have to convert this logical block number to a physical location such as a cylinder, track, and sector of a disk. This is a 64-bit value. The driver should use **b_lblkno** or **b_blkno**, but not both.

b_resid should be set to the number of bytes not transferred because of an error.

b_bufsize contains the size of the allocated buffer.

b_iodone identifies a specific **biodone** routine to be called by the driver when the I/O is complete.

b_error may hold an error code that should be passed as a return code from the driver. **b_error** is set in conjunction with the **B_ERROR** bit set in the **b_flags** member. **bioerror**(9F) should be used in preference to setting the **b_error** field.

b_private is for the private use of the device driver.

b_edev contains the major and minor device numbers of the device accessed.

SEE ALSO

strategy(9E), **bioerror**(9F), **clrbuf**(9F), **getrbuf**(9F), **physio**(9F), **iovec**(9S), **uio**(9S)

Writing Device Drivers

WARNINGS

Buffers are a shared resource within the kernel. Drivers should read or write only the members listed in this section. Drivers that attempt to use undocumented members of the **buf** structure risk corrupting data in the kernel or on the device.