Block Device Driver

Introduction

- Block Device driver is a type of device driver transfers data to and from kernel’s buffer cache.
- Advantage: optimize I/O devices, which handle, (a) data in fixed blocks (b) can store a UNIX file system, by introducing Buffer-cache
- Best examples: Disk drives
- Inappropriate for variable-sized blocks, for example single byte, etc

Test Data generator

Example , to generate data to test a Tape Drive

- *dd if=/dev/blktest of=/dev/tape bs=10k count=1000*
- *tape rewind*
- *dd if=/dev/tape bs=10k | cmp - /dev/blktest*

The Operating System/ Driver Interface

- All entry-points of Character device-driver except read() and write().
- strategy() – combination of both read() and write() entry points.
- strategy() – compulsory entry point for all block device drivers.
- print() – used by kernel to report errors
- All other entry-points are optional.

Internal Operation of a Driver

- Different mechanisms for transferring data to and from driver and rest of the UNIX system.
- I/O requests from the Kernel and not User.
- In Block device-driver, we transfer data to and from kernel's memory space and not User's process area.
- Hence, no need to *copyin* and *copyout* routines.

Contents of Prologue

- #include <sys/types.h>
- #include <sys/cmn_err.h>
- #include <sys/buf.h>

- static char testpattern[1024] = {“0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, ….
  ….. 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff” };

init() entry-point

```
blk1init()
{
   printf(“Test Data Block Driver v1.0\n”);
}
```

open() and close() entry points

- blk1open(dev_t dev, int flag, int id)

```
{
}
```

- blk1close(dev_t dev, int flagm, int id)

```
{
}
```

strategy() entry-point

- Responsible for handling data (in or out)
- Replaces both read() and write() entry points
- Block driver uses a pointer to buffer header in Kernel space rather than u. in User process Area.
- Pointer is passed to the strategy routine
- No concern with process scheduling
- Strategy routine cannot complete I/O, it returns.
- When I/O is completed, block driver notifies Kernel with *iodone(bp),* passing pointer to the processed buffer.

strategy() entry-point

```
strategy entry point:
   if(read request)
    copy data from testpattern to buffer
   set residual count to 0 (indicating a successful transfer)
   report that I/O request has been completed
```

strategy() entry-point

```
blk1strategy(bp)
struct buf *bp;
{
148     if(bp->b_flags & B_READ)
149         bcopy(testpattern, paddr(bp), bp->b_bcount);
150     bp->resid = 0;
151     iodone(bp);
}
```

Single parameter is the pointer to the entry in the buffer cache that is to be read or written driver checks at line 148 , & call bcopy at line 149 to copy the contents of testpattern to the buffer. And driver indicate i/o completed by iodone(bp)

strategy() entry-point

- *bp* : pointer to the entry in buffer cache
- *B_READ* : Variable to check whether it is a read request or write request
- *bcopy()* : Copies the content of *testpattern* to the buffer

  : Method to optimize performance

  : No special functions to transfer data, always succeeds, so no return value.
- *iodone()* : Driver indicates to Kernel that I/O is successfully completed.

The Buffer Cache

- Refer fig. 5.2 pg. 88
- Used by UNIX kernel to store data both incoming and outgoing
- Advantage: Allows user to perform I/O without specifications, improves performance
- Example, User wants to read 1097th record with record-size = 105 bytes
- Starting position = 1097*105 = 115185th byte
- Ending position = 1098*105 – 1 = 115289th byte
- Suppose block-size = 512 bytes
- Then, 115185 / 512 = 224.97th block
- That is: 115199 – 115184 = 15 bytes in Block – 224

  and 115290 – 115200 = 90 bytes in Block – 225

Members of Buf Structure

b_flags	Flags (Information regarding status of the buffer)
b_forw	Pointer used by Kernel to associate block with device
b_back	Pointer used by Kernel to associate block with device
aw_forw	Available list pointer
aw_back	Available list pointer
b_dev	Major and Minor device numbers
b_bcount	Block Count
b_blkno	Block Number
b_sector	Sector Number
b_resid	Bytes not transferred
b_error	Error flags
paddr(bp)	Kernel virtual address of buffer contents

Two important Flags

- B_READ : Which marks a buffer that is to be filled
- B_ERROR : Which the driver sets if the I/O fails
- b_forw and b_back pointers are used by the kernel to keep the buffer on a doubly linked list
- av_forw and av_back are used to maintain available list.
- b_device identifies the major device number
- minor(bp->b_device) identifies the minor device number – (<sys/sysmacros.h>)
- b_bcount : number of bytes to be transferred
- B_SIZE in <sys/param.h> defines the block-size
- b_blkno : location on the disk of data
- b_resid : rest of bytes not copied
- B_ERROR : field in b_flags if set, error occurs
- paddr(bp) : virtual address of the buffer

print() entry point

```
blk1print(dev, message)
dev_t dev;
char *message;
{
cmn_err(CE_WARN, “blk1 driver error: %s\n”, message);
}
```

Used by kernel to report problem related to driver, it only print error message