# A/UX: Block-to-inode Mapping

Revised: 9/30/92
Security: Everyone

A/UX: Block-to-inode Mapping

==========

Article Created: 31 August 1989

Article Change History
---
08/31/92 - REVIEWED
- For technical accuracy.

TOPIC ----------

I am having some problems with a bad block on the disk. I want to 'badblk' it, but I also want to save the file (inode) that used the block to another file before doing that.

I noticed that there is no utility in A/UX 1.1 that does a block-to inode mapping. In SysV systems you usually find icheck, which does this kind of stuff. Did I misread the manuals, or is there no facility to backtrace the blocks to the originated inode? Can I port the icheck to the next release of A/UX?

Another issue with badblk is the so-called "hardware sparing". This is mapped to a ioctl(fd,GD_SPARE),block) inside badblk, which is then translated to a SCSI call in our device driver. I'm using Rodime drives, so I really don't know the hardware-specific way of doing the hardware sparing. I am curious to know how Apple's disks do this. Does Apple support this hardware sparing with our disks? (One could speculate that the disk device copies the block information to another pure block, changes some kind of internal pointer table, and maps the bad block to a bad block table.)

DISCUSSION ----------

We, too, noticed that the 'icheck' utility (block number to i-node number mapping) is not included in A/UX 1.1. The porting of'icheck' utility has not been implemented.

We don't know how well the Rodime hard disk supports the "hardware sparing" either.  However, we believe that the Apple HD80 SC (Quantum Q280) drives are intelligent disk drives and capable of "hardware sparing" support; it incorporates media defect handling and error correcting code capabilities. During regular disk operation, the drive can continue to scan and compensate for any new defective sectors that may show up later on the disks.

On the software side of the bad-blocking (the 'badblk'-set or update bad block information), we have the following test, and it seems to work.

```
# badblk /dev/rdsk/c0d0s31
===> badblk:  block #33988 was not bad blocked
===> badblk:  block #33999 was not bad blocked
===> no blocks bad blocked

# badblk /dev/rdsk/c0d0s31 33988 33999
===> 2 blocks spared, 0 blocks altblked

# badblk /dev/rdsk/c0d0s31
===> no blocks bad blocked
```

According to the description of 'badblk', badblk first attempts to alter a bad block by hardware sparing.  If the hardware sparing fails and the device supports alternate bad blocking, badblk attempts to alternate block the bad block.

Note that for the space-saving reasons, the current A/UX distribution disk partitions do not have the alternate block area reserved for bad block handling.  If the bad sectors occur, it will be done by the hardware sparing.



Tech Info Library Article Number:4344