Supporting large direct-access block devices using 64-bit extensions to the trackdisk.device command set

Ralph Babel <rbabel@babylon.pfm-mainz.de>[*]

15 April 1996

In an effort to overcome the 4-GB limit imposed by the current trackdisk.device interface (which all drivers for direct-access block devices must support in order to be compatible with the standard Amiga disk filesystem), the following set of 64-bit commands has been defined:

- new commands:

command name	decimal value
TD_READ64	24
TD_WRITE64	25
TD_SEEK64	26
TD_FORMAT64	27

  TD_FORMAT64 is provided mostly for reasons of symmetry, as track-oriented formatting is required by very few devices only (e.g. trackdisk.device), and writing a full "track" of data using CMD_WRITE or TD_WRITE64 should not really depend on the previous contents of that track. TD_FORMAT64 is therefore expected to be identical to TD_WRITE64 in most implementations.

- data offset:

  Unlike CMD_READ, CMD_WRITE, TD_SEEK, and TD_FORMAT, all of which deal with a 32-bit-wide byte offset passed via io_Offset, the new commands listed above accept a 64-bit-wide byte offset to address the medium. The lower 32 bits of that byte offset are stored in io_Offset; the upper 32 bits are stored in the io_Actual field, for which the alias io_HighOffset is introduced.

  According to the 1.3 edition of the *RKRM Libraries & Devices*, page 291, a driver may overwrite parameter fields of an I/O request (other than io_Command) in the course of processing a command, so all fields need to be reinitialized upon every command submitted to the driver. io_HighOffset thus does not differ from io_Offset in that regard.

  This way, the io_HighOffset field will not overlap with any of the extra IOExtTD fields required for the ETD-style trackdisk.device commands, and applications can continue to use the original IOStdReq-sized structure. Device-driver programmers are free to implement the TDF_EXTCOM variants of the new 64-bit commands.

  Note: it is perfectly legal for a 64-bit request to cross a 4-GB "boundary" (i.e. $\text{io\_Offset} + \text{io\_Length} > 2^{32}$) or for it to be in 32-bit space completely (i.e. $\text{io\_HighOffset} = 0 \land \text{io\_Offset} + \text{io\_Length} \leq 2^{32}$).

- transfer length:

  io_Length and io_Actual remain 32-bit quantities, and the maximum transfer length is thus still "limited" to $2^{32} - 1$ bytes. With the Amiga's 32-bit address space, this is not a restriction.

[*]This proposal is supported by Dan Babcock <danb@tiac.net>, Randell Jesup <jesup@scala.scala.com>, Oliver Kastl <100447.3361@compuserve.com>, Bernhard Möllemann <zza@mhystic.hall.sub.org>, and Ralph Schmidt <laire@popmail.owl.de>. They all helped tremendously to improve the original draft.

- old commands:

  It is considered an error to use the original 32-bit commands CMD_READ, CMD_WRITE, and TD_FORMAT to access data outside a unit's lower 4-GB data area by "crossing" the upper border of the 32-bit space, i.e. io_Offset plus io_Length *must* be less than or equal to 2^{32}. Drivers are free to reject such commands, to "wrap around" to the beginning (offset 0) of the unit's data area, or to handle them as 64-bit commands with io_HighOffset implicitly set to zero.

  It is therefore recommended that filesystem-level applications use the new 64-bit commands consistently for all accesses to the underlying medium if part of a partition is located outside the data area addressable by the original 32-bit command set. This avoids checks at run-time whether a request extends beyond the 32-bit space.

- compatibility issues:

  Although Commodore-Amiga extended the trackdisk.device command set twice before (OS 1.0: original command set up to TD_PROTSTATUS; OS 1.2: up to TD_REMCHANGEINT; OS 2.0: up to TD_EJECT), it is expected that some drivers do not conform to the standard and provide private commands starting at TD_LASTCOMM or simply crash the system upon receipt of an "out-of-range" command (cf. *RKRM Libraries, 3rd edition*, page 924).

  Therefore, the new 64-bit commands should be sent to a driver if and only if the associated partition they refer to extends beyond the first 4 GB of a unit. This way, a potentially incompatible driver will never receive any of the new commands, and users may safely replace older 32-bit filesystems by 64-bit-compatible updates thereof in existing configurations. Once a filesystem-level application has determined that accessing a partition requires the use of 64-bit commands,[1] it may optionally check whether the underlying driver actually supports them by issuing a TD_READ64 request with io_Offset, io_HighOffset, and io_Length all set to zero; if the underlying driver returns IOERR_NOCMD, the filesystem-level application should report an error (e.g. fail the start-up packet in case of a filesystem). For reasons of efficiency, this check should be performed only *once* upon start-up.

  In deference to older drivers that interpret the new command values differently, 64-bit partitions should not be mounted (be that via a MountList or an RDB entry) on drivers that do not support devices larger than 4 GB. This is no different from the current situation, where most filesystems and drivers will simply "wrap around" and overwrite data in the first 4-GB chunk if a partition extending beyond the first 4 GB of a unit is mounted and written to. This proposal does not intend to address this issue, but neither does it prevent future standards that may allow the caller to identify type and supported features of a device driver.

- alternatives:

  If a driver does not implement the 64-bit commands defined by this proposal, HD_SCSICMD may be used instead to address data above the lower 4-GB range.

- future extensions:

  Implementors of drivers and filesystem-level applications are free to agree on methods to determine whether a driver is at all trackdisk.device-compatible and whether it actually supports the 64-bit command set.

Besides drivers and filesystems, software that needs to be updated to support the extended command set includes Format, DiskCopy, disk-salvage programs, disk editors, and disk reorganizers.

Applications above the filesystem layer, in particular those that make use of the fields id_NumBlocks, id_NumBlocksUsed, and id_BytesPerBlock, may need to be enhanced to be prepared for partitions larger than 4 GB (this includes the CLI command Info).

[1] $(\text{de\_HighCyl} + 1) \times \text{de\_Surfaces} \times \text{de\_BlocksPerTrack} \times \text{de\_SizeBlock} \times 4 > 2^{32}$, i.e. the partition does not reside completely in 32-bit space.