User´s Manual

for

dt - Data Test Program

Version 14.1

by

Robin T. Miller

February 2nd, 2001

EXTREME WARNING!!!

Use of this program is *almost* guaranteed to find problems and cause your schedules to slip. If you are afraid to find bugs or otherwise break your system, then *please* do **not** use this program for testing.

You can pay now or pay later, but you've been warned.


```
The Author of dt,
Robin's Nest Software Inc.      Home e-mail: rmiller@bit-net.com
Robin T. Miller         or      Work e-mail: Robin.Miller@Compaq.com
2 Paradise Lane,                QUICS Login: rtmiller
Hudson, N.H. 03051
Home: (603) 883-2355            Work: (603) 883-2355 (yep, work @ home!)
```

Change History

Version	Date	Author	Changes
14.1	February 2nd, 2001	Robin T. Miller	Added support for better random access device testing via multiple slices option (slices=value), controlling direction via "iodir=" option, variable request sizes (incr=var), setting device block size (dsize=), better support of multiple volumes via "volumes=" and "vrecords=" options. Also updated logic to allow random and reverse I/O to regular disk files.
13.22	November 10th, 2000	Robin T. Miller	Added more test features and options, including: read-after-write (raw), setting the random I/O seed (rseed), and multi-volume media testing.
12.0	July 24, 1999	Robin T. Miller	Add numerous new test features and parameters, including: AIO w/lbdata, AIO w/random, EEI & tape resets, IOT test pattern, larger data/record limits and statistics, Linux & Windows/NT support.
9.3	February 21, 1996	Robin T. Miller	Documented iotype={random or sequential} option.
9.0	December 11, 1995	Robin T. Miller	Logical block data feature, additional (higher) tty speeds, and other minor changes.
8.0	July 26, 1995	Robin T. Miller	Modem testing, child process control, pattern string enhancements, and other minor changes.
7.0	September 11, 1993	Robin T. Miller	Initial release of Users Manual.

1 Overview.

dt is a generic data test program used to verify the proper operation of peripherals & I/O subsystems, and for obtaining performance information. Since verification of data is performed, *dt* can be thought of as a generic diagnostic tool.

Although the original design goals of being a generic test tool were accomplished, it quickly become evident that device specific tests, such as terminals, and different programming interfaces such as memory mapped files and POSIX asynchronous I/O API's were necessary. Therefore, special options were added to enable these test modes and to specify necessary test parameters.

dt command lines are similar to the *dd* program, which is popular on most UNIX systems. *dt* contains numerous options to provide user control of most test parameters so customized tests can be written easily and quickly by specifying simple command line options. Since the exit status of the program always reflects the completion status of a test, scripts can easily detect failures to perform automatic regression tests.

dt has been used to successfully test disks, tapes, serial lines, parallel lines, pipes & FIFO's, memory mapped files, and POSIX Asynchronous I/O. In fact, *dt* can be used with any device that supports the standard open, read, write, and close system calls. Special support is necessary for some devices, such as serial lines, for setting up the speed, parity, data bits, etc, but *dt*'s design provides easy addition of this setup.

Most tests can be initiated by a simple *dt* command line, and lots of I/O can be initiated quickly using multiple processes and/or POSIX AIO, for those operating systems supporing AIO. More complex tests are normally initiated by writing shell scripts and using *dt* in conjunction with other tools, such as *scu* (SCSI Command Utility). Several shell scripts for testing disks, tapes, and serial lines are also supplied with this kit which can used as templates for developing other specialized test scripts.

Specific system features are now being added to *dt* so more extensive testing can be accomplished. The program has been restructured to allow easy inclusion of new device specific tests by dispatching to test functions through a function lookup table. This table gets setup automatically by the program, based on options enabled, or via the device type "*dtype=*" option.

WARNING: *dt* does NOT perform any sanity checking of the output device specified. This means if you are running as 'root' and you specify a raw disk device, *dt* will overwrite existing file systems, so please be careful.

NOTE: Tru64 (Digital) UNIX prevents overwriting the disk label block (block 0), to prevent you from destroying this valuable information. Overwriting other files systems, not starting at block zero, is still possible however.

2 Operating Systems Supported.

dt is conditionalized to run on SUN, ULTRIX, OSF, QNX, Windows/NT, and Linux operating systems. This conditionalization tends to make the source look rather ugly, but I've purposely left this in for code maintainability (common code base) and for other people to see porting differences between these systems. UNIX is NOT as portable as some people think, but the POSIX standard is finally changing this. Eventually this will be cleaned up, but this conditionization is currently necessary so *dt* will compile and run with non-ANSI compliant compilers and non-POSIX compliant operating systems.

2.1 POSIX Compliant Systems.

People who may wish to port *dt* to other POSIX compliant operating systems, should review the Tru64 UNIX, Linux, and QNX conditionalized code. These operating systems also support an ANSI compliant compiler.

3 Test Uses.

Those poeple with an imagination will find many uses for *dt*, but I'll list a few I've used it for, just to whet your appetite:

- Testing of tape devices using different block sizes to determine the best blocking factor for optimum performance and capacity. This is very important for streaming tapes devices.
- Write tapes to end of tape, to determine the total tape capacity. This gives the total data capacity of tapes, after inter-record gaps, preamble/postambles, or pad blocks are written on the tape.
- Read existing tapes with data comparison disabled, to determine the amount of data on the tape. This is useful to determine how much disk space is required to read in a tape, or to simply verify the tape can be read without errors.
- Reading/writing an entire tape to ensure device drivers properly sense and handle end of tape error conditions.
- Write a tape and ensure it can be read on another tape drive to test drive compatibility (also referred to as transportability).
- Read multiple tape files to ensure file marks and end of tape are reported and handled properly by tape drivers.
- I/O to disks using the raw device interface, to determine the optimum performance of the controller. This usually gives a good indication of how well the controller cache or read-ahead improves I/O performance for sequential or random file access.
- I/O to disk files through the file system, to determine the affect the buffer cache has on write and read performance. You must know the characteristics of your O/S's buffer cache to select file sizes to either get optimum performance from the cache, or to defeat the affect of the buffer cache.
- Reading/writing of entire disks, to ensure the media capacity and end of media error handling is properly reported by device drivers.
- Test memory mapped files to compare I/O performance against raw and file system I/O. Typically, memory mapped I/O approaches the raw device performance.
- Testing I/O to files on NFS mounted file systems. This will give you a good indication of your ethernet performance to remote files.
- Writing/reading pipes & FIFO's to verify pipe operation and performance.
- Initiating multiple processes to test optimizations of buffer cache, device drivers, and/or intelligent controllers. This is also useful to test multiple device access and for loading the I/O subsystem.
- Force I/O at different memory boundaries to test low level driver handling. Using the align option, you can set memory alignment for testing specialized device driver DMA code. This is very useful when developing new I/O sub-systems.
- Do loopback testing of parallel or serial lines on either the same system of different systems. This is a useful compatibility test when running different machines running different operating systems.

- Enable POSIX Asynchronous I/O to verify proper operation of this API and to determine performance gains (over standard synchronous I/O). This is also useful for queuing multiple I/O requests to drivers and for testing SCSI tag queuing and RAID configurations.
- Specify variable record options for testing variable tape devices.
- On Tru64 cluster systems, distributed lock manager (DLM) options can be used to control access to shared devices or files.
- Also available on Tru64 UNIX is the ability to use Extended Error Information (EEI) to detect and recover from SCSI bus/device resets (tape is repositioned for continuing the test).

4 Program Options.

This section describes program options and and special notes related to each. The *dt* help file provides a summary of the options, and the default value of most options. The *dt* help summary for Tru64 UNIX is shown in Appendix A.

4.1 Input File "*if=*" Option.

This option specifies the input file to open for reads. The device is opened read-only so devices which only permit or support read access, e.g., parallel input devices, can be opened successfully.

Special Notes:

- Data read is automatically verified with the default data pattern, unless you disable this action via the "*disable=compare*" option.
- Extra pad bytes of sizeof(int), are allocated at the end of data buffers, initialized with the inverted data pattern, and then verified after each read request to ensure the end of data buffers didn’t get overwritten by file system and/or device drivers. This extra check has found problems with flushing DMA FIFO’s on several machines.

```
Syntax:
    if=filename      The input file to read.
```

4.2 Output File "*of=*" Option.

This option specifies the output file to open for writes. After the write portion of the test, the device is closed (to reposition to start of file or to rewind the tape), re-opened, and then a read verification pass is performed. If you wish to prevent the read verify pass, you must specify the "*disable=verify*" option.

Special Notes:

- Terminal devices are **not** closed between passes so previously set terminal characteristics don’t get reset. This also caused a race condition when doing loopback testing with two processes.
- When testing terminal (serial) devices, modem control is disabled (via setting CLOCAL) to prevent tests from hanging. If the "*enable=modem*" option is specified, then CLOCAL is reset, hangup on close HUPCL is set, and testing will not preceed until carrier or DSR is detected. This code is not fully tested, but this description accurately describes the code.
- At the present time, tapes are rewound by closing the device, so you **must** specify the rewind device during testing if the read verify pass is being performed. This restriction will probably change in the next release since magtape control commands will be supported (tape specific tests as well).

- A special check is made for the /dev/ prefix, and if located, the O_CREAT open flag is cleared to prevent accidently creating files in this directory when not specifying the correct device name (very easy to do when running tests as super-user 'root').
- When writing to raw disks on Tru64 UNIX, if the disk was previously labeled, you must issue the "*disklabel -z*" command to destroy the label block or else you cannot write to this area of this disk (block 0). Failure to do this results in the error "Read-only file system" (errno=EROFS) being returned on write requests.

```
Syntax:
    of=filename      The output file to write.
```

4.3 Pattern File "*pf=*" Option.

This option specifies a pattern file to use for the data pattern during testing. This option overrides the "*pattern=*" option and allows you to specify specialized patterns. The only restriction to this option is that the entire file *must* fit in memory. A buffer is allocated to read the entire pattern file into memory before testing starts so performance is not affected by reading the pattern file.

```
Syntax:
    pf=filename      The data pattern file to use.
```

4.4 Block Size "*bs=*" Option.

This option specifies the block size, in bytes, to use during testing. At the present time, this option sets both the input and output block sizes. At the time I originally wrote this program, I didn't have the need for seperate block sizes, but this may change in a future release where I'll add back the "*ibs=*" & "*obs=*" options available with *dd*.

Special Notes:

- When enabling variable length records via the "*min=*" option, this also sets the maximum record size to be written/read.
- For memory mapped files, the block size *must* be a multiple of the system dependent page size (normally 4k or 8k bytes).

```
Syntax:
    bs=value         The block size to read/write.
```

4.5 Log File "*log=*" Option.

This option specifies the log file to redirect all program output to. This is done by re-opening the standard error stream (stderr) to the specifed log file. Since all output from *dt* is directed to stderr, library functions such as perror() also write to this log file.

Special Notes:

- A seperate buffer is allocated for the stderr stream, and this stream is set buffered so timing isn't affected by program output.
- When starting multiple processes via the "*procs=*" option, all output is directed to the same log file. The output from each process is identified by the process ID (PID) as part of the message (errors & statistics).

```
Syntax:
    log=filename     The log file name to write.
```

4.6 POSIX Asynchronous I/O "*aios=*" Option.

This option enables and controls the number of POSIX Asychronous I/O requests used by the program.

Special Notes:

- The default is to queue up to 8 requests.
- This option is only valid for Tru64 UNIX systems are this time.
- The system limit for AIO on Tru64 UNIX is dynamic, and can be queried by using the "*sysconfig -q rt*" command.
- You can use the "*enable=aio*" option to enable AIO and use the default request limit.
- AIO is only supported for character devices and is disabled for terminals. On Tru64 UNIX, you can alter the Makefile and link against libaio.a, which allows AIO with any device/file by mimic'ing AIO using POSIX threads.
- AIO requests can **not** be cancelled on Tru64 UNIX, so queuing many requests to 1/2" tape devices will probably result in running off the end of the tape reel. This is not a problem for cartridge tapes.

```
Syntax:
    aios=value       Set number of AIO's to queue.
```

4.7 Buffer Alignment "*align=*" Option.

This option controls the alignment of the normally page aligned data buffer allocated. This option is often useful for testing certain DMA boundary conditions not easily reproduced otherwise. The rotate option automatically adjust the data buffer pointer by (0, 1, 2, 3, ...) for each I/O request to ensure various boundaries are fully tested.

```
Syntax:
    align=offset     Set offset within page aligned buffer.
or  align=rotate     Rotate data address through sizeof(ptr).
```

4.8 File Disposition "*dispose=*" Option.

This option controls the disposition of test files created on file systems. By default, the test file created is deleted before exiting, but sometimes you may wish to keep this file for further examination, for use as a pattern file, or simply for the read verify pass of another test (e.g., reading the file via memory map API).

```
Syntax:
    dispose=mode     Set file dispose to: delete or keep.
```

4.9 Dump Data Limit "*dlimit=*" Option.

This option allows you to specify the dump data limit used when data compare errors occur. The default dump data limit is 64 bytes.

```
Syntax:
    dlimit=value     Sets the data dump limit to value.
```

4.10 Device Size "*dsize=*" Option.

This option allows you to specify the device block size used. On Tru64 Unix, the device block size is obatined automatically by an OS specific IOCTL. For all other systems, random access devices default to 512 byte blocks. You'll likely use this option with C/DVD's, since their default block size to 2048 bytes per block.

```
Syntax:
    dsize=value      Set the device block (sector) size.
```

4.11 Device Type "*dtype=*" Option.

4.12 Input Device Type "*idtype=*" Option.

4.13 Output Device Type "*odtype=*" Option.

These options provide a method to inform *dt* of the type of device test to be performed. Without this knowledge, only generic testing is possible.

Special Notes:

- On Tru64 UNIX systems, these options are not necessary, since this information is obtained via the DECIOCGET or DEVGETINFO IOCTL's:
- Although the program accepts a large number of device types, as shown below, specific tests only exists for "disk", "tape", "fifo", and "terminal" device types. Others may be added in the future.
- In the case of "disk" device type, *dt* reports the relative block number when read, write, or data compare errors occur.
- Also for "disk" devices, *dt* will automatically determine the disk capacity if a data or record limit is not specified. This is done via a series of seek/read requests.
- On each operating system supported, string compares are done on well known device names to automatically select the device type. For example on QNX, "/dev/hd" for disk, "/dev/tp" for tapes, and "/dev/ser" for serial lines.
- The device type gets displayed in the total statictics.

```
Syntax:
    dtype=string    Sets the device type.
    idtype=string   Sets the input device type.
    odtype=string   Sets the output device type.

The Valid Device Types Are:
    audio     comm      disk      graphics  memory
    mouse     network   fifo      pipe      printer
    processor socket    special   streams   tape
    terminal  unknown
```

Note: Although *dt* does not provide specific test support for each of the devices shown above, its' design makes it easy to add new device specific tests. Specific support exists for disk, fifo, pipe, tape, and terminals. Support for "ptys" may be added in the future as well.

4.14 Error Limit "*errors=*" Option.

This option controls the maximum number of errors tolerated before the program exits.

Special Notes:

- The default error limit is 1.
- All errors have a time stamp associated with them, which may be useful for characterizing intermittent error conditions.
- The error limit is adjusted for read, write, or data compare failures. This limit is not enforced when flushing data, or for certain AIO wait operations which are considered non-fatal (perhaps this will change).
- A future release may support an "*onerr=*" option to control the action of errors (e.g., loop, ignore (continue), or exit).

```
Syntax:
    errors=value     The number of errors to tolerate.
```

4.15 File Limit "*files=*" Option.

This option controls the number of tape files to process with tape devices.

Special Notes:

- During the write pass, a tape file mark is written after each file. After all files are written, 1 or 2 file marks will be written automatically by the tape driver when the device is closed.
- During reads, each file is expected to be terminated by a file mark and read() system calls are expected to return a value of 0 denoting the end of file. When reading past all tapes files, an errno of ENOSPC is expected to flag the end of media condition.
- Writing tape file marks is currently not supported on the QNX Operating System. The release I currently have does not support the mtio commands, and unfortunately the POSIX standard does **not** define this interface (the mtio interface appears to be a UNIX specific standard). Multiple tape files can still be read on QNX systems however.

```
Syntax:
    files=value      Set number of tape files to process.
```

4.16 Terminal Flow Control "*flow=*" Option.

This option specifies the terminal flow control to use during testing.

Special Notes:

- The default flow control is "xon_xoff".
- When using XON/XOFF flow control, you must make sure these byte codes (Ctrl/Q = XON = '\021', Ctrl/S = XOFF = '\023), since the program does not filter these out automatically. Also be aware of terminal servers (e.g., LAT), or modems (e.g., DF296) which may eat these characters.
- Some serial lines do **not** support clear-to-send (CTS) or request-to-send (RTS) modem signals. For example on Alpha Flamingo machines, only one port (/dev/tty00) supports full modem control, while the alternate console port (/dev/tty01) does not. Therefore, if running loopback between both ports, you can not use *cts_rts* flow control, the test will hang waiting for these signals to transition (at least, I think this is the case).

```
Syntax:
    flow=type        Set flow to: none, cts_rts, or xon_xoff.
```

4.17 Record Increment "*incr=*" Option.

This option controls the bytes incremented when testing variable length records. After each record, this increment value (default 1), is added to the last record size (starting at "*min=*", up to the maximum record size "*max=*").

Special Notes:

- If variable length record testing is enabled on fixed block disks and this option is omitted, then "*incr=* defaults to 512 bytes.

```
Syntax:
    incr=value       Set number of record bytes to increment.
or  incr=variable    Enables variable I/O request sizes.
```

4.18 I/O Direction "*iodir=*" Option.

This option allows you to control the I/O direction with random access devices. The default direction is forward.

```
Syntax:
    iodir=direction  Set I/O direction to: {forward or reverse}.
```

4.19 I/O Mode "*iomode=*" Option.

This option controls the I/O mode used, either copy, test, or verify modes. The copy option was added to do a byte for byte copy between devices, while skipping bad blocks and keeping file offsets on both disks in sync. I've used this option to (mostly) recover my system disk which developed bad blocks which could not be re-assigned. A verify operation automatically occurs after the copy, which is real handy for unreliable diskettes.

```
Syntax:
    iomode=mode      Set I/O mode to: {copy, test, or verify}.
```

4.20 I/O Type "*iotype=*" Option.

This option controls the type of I/O performed, either random or sequential. The default is to do sequential I/O.

Special Notes:

- The random number generator used is chosen by defines: RAND48 to select srand48()/lrand48(), RANDOM to select srandom()/random(), and if neither are defined, srand()/rand() gets used by default. Refer to your system literature or manual pages to determine which functions are supported.

```
Syntax:
    iotype=type        Set I/O type to: {random or sequential}.
```

The seeks are limited to the data limited specified or calculated from other options on the *dt* command line. If data limits are not specified, seeks are limited to the size of existing files, or to the entire media for disk devices (calculated automatically by *dt*). If the data limits exceed the capacity of the media/partition/file under test, a premature end-of-file will be encountered on reads or writes, but this is treated as a warning (expected), and not as an error.

4.21 Minimum Record Size "*min=*" Option.

This option controls the minimum record size to start at when testing variable length records.

Special Notes:

- By default, *dt* tests using fixed length records of block size "*bs=*" bytes.
- This option, in conjuntion with the "*max=*" & "*incr=*" control variable length record sizes.
- If variable length record testing is enabled on fixed block disks and this option is omitted, then "*min=* defaults to 512 bytes.

```
Syntax:
    min=value        Set the minumum record size to transfer.
```

4.22 Maxmimum Record Size "*max=*" Option.

The option controls the maximum record size during variable length record testing.

Special Notes:

- If the "*min=*" option is specified, and this option is omitted, then the maximum record size is set to the block size "*bs=*".
- This option, in conjuntion with the "*min=*" & "*incr=*" control variable length record sizes.

```
Syntax:
    max=value        Set the maximum record size to transfer.
```

4.23 Logical Block Address "*lba=*" Option.

This option sets the starting logical block address used with the *lbdata* option. When specified, the logical block data (*enable=lbdata*) option is automatically enabled.

```
Syntax:
    lba=value        Set starting block used w/lbdata option.
```

Special Notes:

- Please do not confuse this option with the disks' real logical block address. See *dt*'s "*seek=*" or "*position=*" options to set the starting file position.
- Also note that *dt* doesn't know about disk partitions, so any position specified is relative to the start of the partition used.

4.24 Logical Block Size "*lbs=*" Option.

This option sets the starting logical block size used with the *lbdata* option. When specified, the logical block data (*enable=lbdata*) option is automatically enabled.

```
Syntax:
    lbs=value        Set logical block size for lbdata option.
```

4.25 Data Limit "*limit=*" Option.

This option specifies the number of data bytes to transfer during each write and/or read pass for the test.

Special Notes:

- You must specify either a data limit, record limit, or files limit to initiate a test, unless the device type is "disk", in which case *dt* will automatically determine the disk capacity.
- When specifying a runtime via the "*runtime=*" option, the data limit controls how many bytes to process for each pass (write and/or read pass).
- If you specify a infinite "*limit=Inf*" value, each pass will continue until the end of media or file is reached.

```
Syntax:
    limit=value      The number of bytes to transfer.
```

4.26 Munsa (DLM) "*munsa=*" Option.

This option is used on Tru64 Cluster systems to specify various distributed lock manager (DLM) options with devices or files.

```
Syntax:
    munsa=string     Set munsa to: cr, cw, pr, pw, ex.

MUNSA Lock Options:
    cr = Concurrent Read (permits read access, cr/pr/cw by others)
    pr = Protected Read (permits cr/pr read access to all, no write)
    cw = Concurrent Write (permits write and cr access to resource by all)
    pw = Protected Write (permits write access, cr by others)
    ex = Exclusive Mode (permits read/write access, no access to others)

        For more details, please refer to the dlm(4) reference page.
```

Special Notes:

- MUNSA is an obsolete Tru64 Cluster term which meant *MUltiple Node Simultaneous Access.* The new term is DAIO for *Direct Access I/O.*

4.27 Common Open Flags "*flags=*" Option.

4.28 Output Open Flags "*oflags=*" Option.

These options are used to specify various POSIX compliant open flags, and system specific flags, to test the affect of these open modes.

Special Notes:

- Each operating system has different flags, which can be queried by reviewing the *dt* help text ("*dt help*").

```
Syntax:
    flags=flags      Set open flags: {excl,sync,...}.
    oflags=flags     Set output flags: {append,trunc,...}.
```

4.29 On Child Error "*oncerr=*" Option.

This option allows you to control the action taken by *dt* when a child process exits with an error. By default, the action is *continue*, which allows all child processes to run to completion. If the child error action is set to *abort*, then *dt* aborts all child processes if **any** child process exits with an error status.

```
    Syntax:
        oncerr={abort|continue}  Set child error action.
```

4.30 Terminal Parity Setting "*parity=*" Option.

This option specifies the terminal parity setting to use during testing.

```
    Syntax:
        parity=string    Set parity to: even, odd, or none.
 on QNX parity=string    Set parity to: even, odd, mark, space, or none.
```

4.31 Pass Limit "*passes=*" Option.

This option controls the number of passes to perform for each test.

Special Notes:

- The default is to perform 1 pass.
- When using the "*of=*" option, each write/read combination is considered a single pass.
- When multiple passes are specified, a different data pattern is used for each pass, unless the user specified a data pattern or pattern file. [Please keep this in mind when using the "*dispose=keep*" option, since using this same file for a subsequent *dt* read verify pass, will report comparison errors... I've burned myself this way.]

```
    Syntax:
        passes=value     The number of passes to perform.
```

4.32 Data Pattern "*pattern=*" Option.

This option specifies a 32 bit hexadecimal data pattern to be used for the data pattern. *dt* has 12 built-in patterns, which it alternates through when running multiple passes. The default data patterns are:

```
   0x39c39c39, 0x00ff00ff, 0x0f0f0f0f, 0xc6dec6de, 0x6db6db6d, 0x00000000,
   0xffffffff, 0xaaaaaaaa, 0x33333333, 0x26673333, 0x66673326, 0x71c7c71c
```

You can also specify the special keyword "*incr*" to use an incrementing data pattern, or specify a character string (normally contained within single or double quotes).

```
    Syntax:
        pattern=value    The 32 bit hex data pattern to use.
    or  pattern=iot      Use DJ's IOT test pattern.
    or  pattern=incr     Use an incrementing data pattern.
    or  pattern=string   The string to use for the data pattern.
```

So, what is DJ's IOT test pattern? This pattern places the logical block address (lba) in the first word (4 bytes) of each block, with (lba+=0x01010101) being placed in all remaining words in the data block (512 bytes by default). In this way, the logical block is seeded throughout each word in the block.

When specifying a pattern string via "*pattern=string*", the following special mapping occors:

```
Pattern String Mapping:
   \\ = Backslash   \a = Alert (bell)   \b = Backspace
   \f = Formfeed    \n = Newline        \r = Carriage Return
   \t = Tab         \v = Vertical Tab   \e or \E = Escape
   \ddd = Octal Value    \xdd or \Xdd = Hexadecimal Value
```

4.33 File Position "*position=*" Option.

This option specifies a byte offset to seek to prior to starting each pass of each test.

```
Syntax:
    position=offset  Position to offset before testing.
```

4.34 Multiple Processes "*procs=*" Option.

This option specifies the number of processes to initiate performing the same test. This option allows an easy method for initiating multiple I/O requests to a single device or file system.

Special Notes:

- The per process limit on Tru64 UNIX is 64, and can be queried by using the "*sysconfig -q proc*" command.
- Spawning many processes can render your system useless, well at least very slow, and consumes large amounts of swap space (make sure you have plenty!).
- The parent process simply monitors (waits for) all child prcoesses.
- When writing to a file system, the process ID (PID) is appending to the file name specified with the "*of=*" option to create unique file names. If no pattern is specified, each process is started with a unique data pattern. Subsequent passes cycle through the 12 internal data patterns. Use "*disable=unique*" to avoid this new (Version 14.1) behaviour.
- The spawn() facility, used to execute on a different node, is not implemented on the QNX Operating System at this time.

```
Syntax:
    procs=value      The number of processes to create.
```

4.35 Random I/O Offset Alignment "*ralign=*" Option.

This option is used when performing random I/O, to align each random block offset to a particular alignment, for example 32K.

```
Syntax:
    ralign=value     The random I/O offset alignment.
```

4.36 Random I/O Data Limit "*rlimit=*" Option.

This option is used with random I/O to specify the number of bytes to limit random I/O between (starting from block 0 to this range). This option is independent of the data limit option.

```
Syntax:
    rlimit=value     The random I/O data byte limit.
```

4.37 Random Seed Value *rseed=*" Option.

This options sets the seed to initialize the random number generator with, when doing random I/O. When selecting random I/O, the total statistics displays the random seed used during that test. This option can be used to repeat the random I/O sequence of a test.

```
Syntax:
    rseed=value      The random seed to initialize with.
```

4.38 Record Limit "*records=*" Option.

This option controls the number of records to process for each write and/or read pass of each test. The "*count=*" option is an alias for this option (supported for *dd* compatibility).

Special Notes:

- You must specify either a data limit, record limit, or files limit to initiate a test, unless the device type is "disk", in which case *dt* will automatically determine the disk capacity.
- When specifying a runtime via the "*runtime=*" option, the record limit controls how many records process for each pass (write and/or read pass).
- If you specify a infinite "*records=Inf*" value, each pass will continue until the end of media or file is reached.

```
Syntax:
    records=value    The number of records to process.
```

4.39 Run Time "*runtime=*" Option.

This option controls how long the total test should run. When used in conjunction with a data limit or record limit, multiple passes will be performed until the runtime limit expires. A later section entitled "*Time Input Parameters*", describes the shorthand notation for time values.

```
Syntax:
    runtime=time     The number of seconds to execute.
```

4.40 Slices "*slices=*" Option.

This option is used with random access devices. This option divides the media into slices. Each slice contains a different range of blocks to operate on in a separate process. If no pattern is specified, then each slice is started with a unique data pattern. Subsequent passes alternate through *dt's* 12 internal patterns.

```
Syntax:
    slices=value     The number of disk slices to test.
```

Note: This option can be used in conjuntion with multiple processes and/or asynchronous I/O options to generate a heavy I/O load, great for stress testing!

4.41 Record Skip "*skip=*" Option.

This option specifies the numer of records to skip prior to starting each write and/or read pass of each test. The skips are accomplished by reading records.

```
Syntax:
    skip=value       The number of records to skip past.
```

4.42 Record Seek "*seek=*" Option.

This option specifies the number of records to seek past prior to starting each write and/or read test. The seeks are accomplished by lseek()'ing past records, which is much faster than skipping when using random access devices (*dd* could use this option).

```
Syntax:
    seek=value       The number of records to seek past.
```

4.43 Data Step "*step=*" Option.

This option is used to specify non-sequential I/O requests to random access devices. Normally, *dt* does sequential read & writes, but this option specifies that step bytes be seeked past after each request.

```
Syntax:
    step=value       The number of bytes seeked after I/O.
```

4.44 Terminal Speed "*speed=*" Option.

This option specifies the terminal speed (baud rate) to setup prior to initiating the test. Although *dt* supports all valid baud rates, some speeds may not be supported by all serial line drivers, and in some cases, specifying higher speeds may result in hardware errors (e.g., silo overflow, framing error, and/or hardware/software overrun errors). The valid speeds accepted by *dt* are:

```
       0        50        75       110       134       150
     200       300       600      1200      1800      2400
    4800      9600     19200     38400      57600   115200
```

Although a baud rate of zero is accepted, this is done mainly for testing purposes (some systems use zero to hangup modems). The higher baud rates are only valid on systems which define the Bxxxxx speeds in termios.h.

Special Notes:
- The default speed is 9600 baud.

```
Syntax:
    speed=value      The tty speed (baud rate) to use.
```

4.45 Terminal Read Timeout "*timeout=*" Option.

This option specifies the timeout to use, in 10ths of a second, when testing terminal line interfaces. This is the timeout used between each character after the first character is received, which may prevent tests from hanging when a character is garbled and lost.

Special Notes:
- The default terminal timeout is 3 seconds.
- The default timeout is automatically adjusted for slow baud rates.

```
Syntax:
    timeout=value    The tty read timeout in .10 seconds.
```

4.46 Terminal Read Minimum "*ttymin=*" Option.

This option specifies the minmum number of characers to read, sets the VMIN tty attribute.

Special Notes:

- The tty VMIN field normally gets sets to the value of the block size (*bs=balue*).
- Note that on some systems, the VMIN field is an *unsigned char*, so the maximum value is 255.
- On QNX, this field is an *unsigned short*, so a maximum of 65535 is valid.

```
Syntax:
    ttymin=value     The tty read minimum count (sets vmin).
```

4.47 Multiple Volumes "*volumes=*" Option.

4.48 Multi-Volume Records "*vrecords=*" Option.

These options are used with removal media devices, to define how many volumes and records on the last volume to process (i.e., tapes, etc). By using these options, you do not have to *guess* at a data limit or record limit, to overflow onto subsequent volumes. These options automatically sets the "*enable=multi*" option.

```
Syntax:
    volumes=value    The number of volumes to process.
    vrecords=value   The record limit for the last volume.
```

4.49 Enable "*enable=*" & Disable *disable=* Options.

These options are used to either enable or disable program flags which either alter default test modes, test actions, or provide additional debugging information. You can specify a single flag or multiple flags each seperated by a comma (e.g., "*enable=aio,debug,dump*").

```
Syntax:
    enable=flag      Enable one or more of the flags below.
    disable=flag     Disable one or more of the flags below.
```

The flags which can be enabled or disabled are described below.

4.49.1 POSIX Asynchronous I/O "*aio*" Flag.

This flag is used to control use of POSIX Asynchronous I/O during testing, rather than the synchronous I/O read() and write() system calls.

Special Notes:

- This test mode is only supported on the Tru64 UNIX Operating System at this time.
- Beware, you may need to rebuild *dt* on new versions of Tru64 Unix due to POSIX changes and/or AIO library changes between major releases.
- Reference the "*aios=*" option, for more special notes.

```
Flag:
    aio              POSIX Asynchronous I/O.(Default: disabled)
```

4.49.2 Reporting Close Errors "*cerror*" Flag.

This flag controls where close errors are reported as an error or a failure. When disabled, close errors are reported as a warning. This flag is meant to be used as a workaround for device drivers which improperly return failures when closing the device. Many system utilities ignore close failures, but when testing terminals and tapes, the close status us *very* important. For example with tapes, the close reflects the status of writing filemarks (which also flush buffered data), and the rewind status.

```
Flag:
    cerrors          Report close errors.   (Default: enabled)
```

4.49.3 Data Comparison "*compare*" Flag.

This flag disables data verification during the read pass of tests. This flag should be disabled to read to end of file/media to obtain maximum capacity statistics, or to obtain maximum performance statistics (less overhead).

```
Flag:
    compare          Data comparison.       (Default: enabled)
```

4.49.4 Core Dump on Errors "*coredump*" Flag.

This flag controls whether a core file is generated, via abort(), when *dt* is exiting with a failure status code. This is mainly used for program debug, and is not of much interest to normal users. When testing multiple processes, via fork(), this is the only way to debug since the standard *dbx* debugger can't be used with child processes (this is finally changing in the Tru64 UNIX V2.0 release, we've waited a long time).

```
Flag:
    coredump         Core dump on errors.   (Default: disabled)
```

4.49.5 Diagnostic Logging "*diag*" Flag.

This option is only valid on Tru64 Unix. When enabled, error messages get logged to the binary error logger. This is useful to correlate device error entries with test failures. Please note, the logging only occurs when running as superuser (API restriction, not mine!).

```
Flag:
    diag             Log diagnostic msgs.   (Default: disabled)
```

4.49.6 Debug Output "*debug*" Flag.

4.49.7 Verbose Debug Output "*Debug*" Flag.

4.49.8 Random I/O Debug Output "*rdebug*" Flag.

These flags enable two different levels of debug, which are useful when trouble-shooting certain problems (i.e., what is *dt* doing to cause this failure?). Both flags can be specified for full debug output.

```
Flag:
    debug            Debug output.          (Default: disabled)
    Debug            Verbose debug output.  (Default: disabled)
    rdebug           Random debug output.   (Default: disabled)
```

4.49.9 Dump Data Buffer "*dump*" Flag.

This flag controls dumping of the data buffer during data comparision failures. If a pattern file is being used, then the pattern buffer is also dumped for easy comparision purposes. To prevent too many bytes from being dumped, esp. when using large block sizes, dumping is limited to 64 bytes of data.

Special Notes:

- When the failure occurs within the first 64 bytes of the buffer, dumping starts at the beginning of the buffer.
- When the failure occurs at some offset within the data buffer, then dumping starts at (data limit/2) bytes prior to the failing byte to provide context.
- The start of the failing data is marked by an asterisk '*'.
- You can use the *dlimit=* option to override the default dump limit.
- Buffer addresses are displayed for detection of memory boundary problems.

```
Flag:
    dump             Dump data buffer.      (Default: enabled)
```

4.49.10 Tape EEI Reporting "*eei*" Flag.

This option controls the reporting of Extended Error Information (EEI) on Tru64 UNIX systems, for tape devices when errors occur. The standard tape information available from *mt* is reported, along with the EEI status, CAM status, and SCSI request sense data. This is excellent information to help diagnose tape failures. (thank-you John Meneghini!)

```
Flag:
    eei              Tape EEI reporting.    (Default: enabled)
```

4.49.11 Tape Reset Handling "*resets*" Flag.

This option is used during SCSI bus and device reset testing, to reposition the tape position (tapes rewind on resets), and to continue testing. This option is only enabled for Tru64 UNIX systems (currently), since this option requires reset detection from EEI status, and tape position information from the CAM tape driver (although *dt* also maintains the tape position as a sanity check against the drivers' data).

```
Flag:
    resets           Tape reset handling.   (Default: disabled)
```

4.49.12 Flush Terminal I/O Queues "*flush*" Flag.

This flag controls whether the terminal I/O queues get flushed before each test begins. This must be done to ensure no residual characters are left in the queues from a prior test, or else data verification errors will be reported. Residual characters may also be left from a previous XOFF'ed terminal state (output was suspended).

```
Flag:
    flush            Flush tty I/O queues.  (Default: enabled)
```

4.49.13 Log File Header "*header*" Flag.

When a log file is specified, *dt* automatically writes the command line and *dt* version information at the beginning of the log file. This option allows you to control whether this header should be written.

```
Flag:
    header           Log file header.       (Default: enabled)
```

4.49.14 Logical Block Data Mode "*lbdata*" Flag.

This option enables a feature called logical block data mode. This feature allows reading/writing of a 4-byte (32-bit) logical block address at the beginning of each data block tested. The block number is stored using SCSI byte ordering (big-endian), which matches what the SCSI Write Same w/lbdata option uses, so *dt* can verify this pattern, generated by *scu*'s "write same" command.

Special Notes:

- The starting logical block address defaults to 0, unless overridden with the "*lba=*" option.
- The logical block size defaults to 512 bytes, unless overridden with the "*lbs=*" option.
- The logical block address is always inserted started at the beginning of each data block (record).
- Enabling this feature will degrade performance statistics (slightly).

4.49.15 Enable Loopback Mode "*loopback*" Flag.

This flag specifies that either the input or output file should be used in a loopback mode. In loopback mode, *dt* forks(), and makes the child process the reader, while the parent process becomes the writer. In previous versions of *dt*, you had to specify both the same input and output file to enable loopback mode. When specifying this flag, *dt* automatically duplicates the input or output device, which is a little cleaner than the old method (which still works).

Some people may argue that *dt* should automatically enable loopback mode when a single terminal or FIFO device is detected. The rationale behind not doing this is described below:

1. You may wish to have another process as reader and/or writer (which also includes another program, not necessarily *dt*).
2. You may wish to perform device loopback between two systems (e.g., to verify the terminal drivers of two operating systems are compatible).
3. A goal of *dt* is *not* to force (hardcode) actions or options to make the program more flexible. A minimum of validity checking is done to avoid being too restrictive, although hooks exists to do this.

Special Notes:

- The read verify flag is automatically disabled.
- This mode is most useful with terminal devices and/or FIFO's (named pipes).

4.49.16 Microsecond Delays "*microdelay*" Flag.

This flag tells *dt* that delay values, i.e. "*sdelay=*" and others, should be executed using microsecond intervals, rather the second intervals. (thank-you George Bittner for implementing this support!)

```
Flag:
    microdelay       Microsecond delays.    (Default: disabled)
```

4.49.17 Memory Mapped I/O "*mmap*" Flag.

This flag controls whether the memory mapped API is used for testing. This test mode is currently supported on SUN/OS, Tru64 UNIX, and Linux operating systems.

Special Notes:

- The block size specified "*bs=*" *must* be a multiple of the system dependent page size (normally 4k or 8k).
- An msync() is done after writing and prior to closing to force modified pages to permanent storage. It may be useful to add an option to inhibit this action at some point, but my testing was specifically to time mmap performance. Obviously, invalidating the memory mapped pages, kind of defeats the purpose of using memory mapped files in the first place.
- Specifying multiple passes when doing a read verify test, gives you a good indication of the system paging utilization on successive passes.
- Memory mapping large data files (many megabytes) may exhaust certain system resources. On an early version of SUN/OS V4.0?, I could hang my system by gobbling up all of physical memory and forcing paging (this was certainly a bug which has probably been corrected since then).

```
Flag:
    mmap             Memory mapped I/O.     (Default: disabled)
```

4.49.18 Test Modem Lines "*modem*" Flag.

This flag controls the testing of terminal modem lines. Normally, *dt* disables modem control, via setting CLOCAL, to prevent tests from hanging. When this flag is enabled, *dt* enables modem control, via clearing CLOCAL, and then monitoring the modem signals looking for either carrier detect (CD) or dataset ready (DSR) before allowing the test to start.

Special Notes:

- The program does not contain modem signal monitoring functions for the all operating systems. The functions in *dt* are specific to Tru64 UNIX and ULTRIX systems, but these can be used as templates for other operating systems.

```
Flag:
    modem            Test modem tty lines.  (Default: disabled)
```

,nh 3 Multiple Volumes "*multi*" Flag. This flag controls whether multiple volumes are used during testing. When this flag is enabled, if the data limit or record count specified does not fit on the current loaded media, the user is prompted to insert the next media to continue testing. Although this is used mostly with tape devices, it can be used with any removeable media.

```
Flag:
    multi            Multiple volumes.      (Default: disabled)
```

4.49.19 Control Per Pass Statistics "*pstats*" Flag.

This flag controls whether the per pass statistics are displayed. If this flag is disabled, a single summary line is still displayed per pass and the total statistics are still displayed in the full format.

```
Flag:
    pstats           Per pass statistics.   (Default: enabled)
```

4.49.20 Read After Write "*raw*" Flag.

This flag controls whether a read-after-write will be performed. Sorry, *raw* does **not** mean character device interface. Normally *dt* performs a write pass, followed by a read pass. When this flag is enabled the read/verify is done immediately after the write.

```
Flag:
    raw              Read after write.      (Default: disabled)
```

4.49.21 Control Program Statistics "*stats*" Flag.

This flag controls whether any statistics get displayed (both pass and total statistics). Disabling this flag also disabled the pass statistics described above.

```
Flag:
    stats            Display statistics.    (Default: enabled)
```

4.49.22 Table(sysinfo) timing "*table*" Flag.

On Tru64 UNIX systems, this option enables additional timing information which gets reported as part of the statistics display. (thanks to Jeff Detjen for adding this support!)

```
Flag:
    table            Table(sysinfo) timing. (Default: disabled)
```

4.49.23 Unique Pattern "*unqiue*" Flag.

This flag controls whether multiple process, get a unqiue data pattern. This affects processes started with the "*slices=*" or the "*procs=*" options. This only affects the *procs=* option when writing to a regular file.

```
Flag:
    unique           Unique pattern.        (Default: enabled)
```

4.49.24 Verbose Output "*verbose*" Flag.

This flag controls certain informational program messages such as reading and writing partial records. If you find these messages undesirable, then they can be turned off by disabling this flag.

```
Flag:
    verbose          Verbose output.        (Default: enabled)
```

4.49.25 Verify Data "*verify*" Flag.

This flag controls whether the read verify pass is performed automatically after the write pass. Ordinarily, when specifying an output device via the "*of=*" option, a read verify pass is done to read and perform a data comparision. If you only wish to write the data, and omit the data verification read pass, then disable this flag.

```
Flag:
    verify           Verify data written.    (Default: enabled)
```

Special Notes:

- If you don't plan to ever read the data being written, perhaps for performance reasons, specifying "*disable=compare*" prevents the data buffer from being initialized with a data pattern.
- This verify option has no affect when reading a device. You must disable data comparsions via "*disable=compare*".

4.50 Program Delays.

dt allows you to specify various delays to use at certain points of the test. These delays are useful to slow down I/O requests or to prevent race conditions when testing terminals devices with multiple processes, or are useful for low level driver debugging. All delay values are in seconds, unless you specify "*enable=microdelay*", to enable micro-second delays.

4.50.1 Close File "*cdelay=*" Delay.

This delay, when enabled, is performed prior to closing a file descriptor.

```
Delay
    cdelay=value     Delay before closing the file.    (Def: 0)
```

4.50.2 End of Test "*edelay=*" Delay.

This delay, when enabled, is used to delay after closing a device, but prior to re-opening the device between multiple passes.

```
Delay
    edelay=value     Delay between multiple passes.    (Def: 0)
```

4.50.3 Read Record "*rdelay=*" Delay.

This delay, when enabled, is used prior to issuing each read request (both synchronous read()'s and asynchronous aio_read()'s).

```
Delay
    rdelay=value     Delay before reading each record. (Def: 0)
```

4.50.4 Start Test "*sdelay=*" Delay.

This delay, when enabled, is used prior to starting the test. When testing terminal devices, when not in self loopback mode, the writing process (the parent) automatically delays 1 second, to allow the reading process (the child) to startup and setup its' terminal characteristics. If this delay did not occur prior to the first write, the reader may not have its' terminal characteristics (flow, parity, & speed) setup yet, and may inadvertantly flush the writers data or receive garbled data.

```
Delay
    sdelay=value     Delay before starting the test.   (Def: 0)
```

4.50.5 Child Terminate "*tdelay=*" Delay.

This delay is used by child processes before exiting, to give the parent process sufficient time to cleanup and wait for the child. This is necessary since if the child exits first, a SIGCHLD signal may force the parent to it's termination signal handler before it's ready to. This is a very simplistic approach to prevent this parent/child race condition and is only currently used by the child for terminal loopback testing.

```
Delay
    tdelay=value     Delay before child terminates.    (Def: 1)
```

4.50.6 Write Record "*wdelay=*" Delay.

This delay, when enabled, is used prior to issuing each write request (both synchronous write()'s and asynchronous aio_write()'s).

```
Delay
    wdelay=value     Delay before writing each record. (Def: 0)
```

4.51 Numeric Input Parameters

For any options accepting numeric input, the string entered may contain any combination of the following characters:

```
Special Characters:
    w = words (4 bytes)            q = quadwords (8 bytes)
    b = blocks (512 bytes)         k = kilobytes (1024 bytes)
    m = megabytes (1048576 bytes)  p = page size (8192 bytes)
    g = gigabytes (1073741824 bytes)
    t = terabytes (1099511627776 bytes)
    inf or INF = infinity (18446744073709551615 bytes)

Arithmetic Characters:
    + = addition                   - = subtraction
    * or x = multiplcation         / = division
    % = remainder

Bitwise Characters:
    ~ = complement of value       >> = shift bits right
   << = shift bits left            & = bitwise 'and' operation
    | = bitwise 'or' operation     ^ = bitwise exclusive 'or'
```

The default base for numeric input is decimal, but you can override this default by specifying 0x or 0X for hexadecimal coversions, or a leading zero '0' for octal conversions.

NOTE: Certain values will vary depending on the operating system and/or machine you are running on. For example, the page size is system dependent, and the value for Infinity is the largest value that will fit into an unsigned long (value shown above is for 64-bit systems), or double for systems which don't support "*long long*".)

4.52 Time Input Parameters.

When specifying the run time "*runtime=*" option, the time string entered may contain any combination of the following characters:

```
Time Input:
    d = days (86400 seconds),      h = hours (3600 seconds)
    m = minutes (60 seconds),      s = seconds (the default)
```

Arithmetic characters are permitted, and implicit addition is performed on strings of the form '1d5h10m30s'.

5 Future Enhancements.

Although many system dependent tests could be added to *dt*, my preference is to add tests for features standard on most systems. I have to admit though, recently I've added many Tru64 UNIX features, required for advanced testing.

There's alot of *dt* features and flags nowadays, so maybe I'll add an initialization file (*.dtrc*) to control certain defaults. We'll see.

A future release may permit *dt* to support multiple devices simlutaneously, and permit dynamic modification of test parameters and obtaining snap-shot statistics. This will require an extensive re-write though, so this isn't likely to happen too soon. I'd like *dt* to be like the RSX-11M *iox* program, which was really nifty. But of course, an operating system which supports asynchronous I/O and event notification, makes this much easier.

Lots of people prefer window based applications nowadays. If I get around to this, it'll probably be a Tcl/Tk/Wish wrapper, or something that will run with your favorite browser. I like this latter idea for implementing remote testing. A native Windows/NT version will most likely come about one day, unless Cygnus Solutions implements POSIX AIO and Alpha hardware in a future release, or Microsoft releases better POSIX API support.

There's a fair amount of work necessary to do extensive tape testing, due to the number of special tape commands (both standard and optional), and differences between tape vendors and operating system API's.

I'm also considering writing a script processor, kinda like a maintenance program generator, to help with this effort so customized tests can be written, but perhaps this is best done with existing tools available (I'm not into reinventing the wheel).

6 Final Comments.

I'm happy to report that *dt* is getting wide spread use here at Compaq. The storage groups, terminal/lat groups, Q/A, developers, and other peripheral qualification groups are all using *dt* as part of their testing (just when I was about to give up on this place... maybe there's hope yet :-).

Anyways, I hope you find *dt* as useful as I have. This is usually one of the first tools I port to a new operating system, since it's an excellent diagnostic and performance tool (it gives me a warm and fuzzy feeling).

Please send me mail on any problems or suggestions you may have, and I'll try to help you out. The future development of *dt* depends alot on user interest. Many of *dt*'s features have come about from user requests.

If You Like My Work,
You Can Do
One Of Two Things:

THROW MONEY or **APPLAUD***

***I've Heard Enough Applause!**

APPENDIX A
dt Help Text

The following help text is contained within the *dt* program. The newest features are highlighted below, as well as previous features for reference by former/existing *dt* users.

The following changes have been made to 'dt' Version 12.0:

- Support for tape Extended Error Information (EEI) and SCSI bus or device reset recovery has been added (to reposition the tape). The EEI support is enabled by default, but the reset support is disabled by default (to avoid breaking existing test scripts). The reset option can also be used w/POSIX Asynchronous I/O (AIO). NOTE: EEI support is Digital UNIX specific (MTIOCGET extension). [Use the "enable/disable={eei,resets}" option to control these.]
- The logical block data feature, "enable=lbdata" option, has been extended so it can be used with random I/O, "iotype=random" option.
- Two new options were added for better range control of random I/O.

  The "ralign=value" option forces block alignment to a byte offset. For example, "ralign=32b" aligns each random request to 32 blocks.

  The "rlimit=value" option contols the upper random I/O data limit. If rlimit isn't specified, it defaults to the data limit, or to the entire disk if no limits are specified.
- Changes to cluster DLM code were merged from work by George Bittner. George also enhanced the random I/O code so the random number is taken as a block, rather than a byte offset, which provides better randomness for large disks or files. (Thanks George!)
- Other changes and/or bug fixes:

```
- When specifying a runtime, ensure we break out of the passes
  loop if we exceed the error limit.  Previously, we'd loop
  (possibly with  error) for the duration of the runtime.
- Fix incorrect record number displayed when Debug is enabled.
- Don't exit read/write loops when processing partial records.
- Fix problem in write function, where short write processing,
  caused us not to write sufficent data bytes (actually, the
  file loop in write_file() caused dtaio_write_data() to be
  called again, and we'd actually end up writing too much!
- When random I/O and lbdata options are both enabled, use the
  file offset seeked to as the starting lbdata address.
- Fix problem reporting total files processed count when we have
  not completed a pass (exiting prematurely due to error or signal).
```

- *dt* compiles and runs on MS Windows 95/98/NT using Cygnus Solution GNU tools. Use Makefile.win32 for compiling on Windows. [a.k.a. GNU-Win32, URL: http://sourceware.cygnus.com/cygwin/]
- When directing output to a log file, "log=LogFile" option, the command line and version string are also emitted.

- The number of I/O per second are now reported in statistics.
- For Linux and Windows/NT, larger data/record limits and statistics are now possible using either "long long" or "double" storage.
- A new data pattern option was added "*pattern=iot*", which encoded the logical block address throughout each word in the data block. Thanks to DJ Brown for this logic from IOT.

The following changes have been made to *dt* Version 9.0:

- Fixed a problem checking pad bytes when doing long reads of short records (this was discovered by someone testing tapes).
- For systems which support higher serial line speeds, you can now select baud rates of 57600, 76800, and 115200. Platinum (ptos) now supports 57600 and 115200 (thank-you Dennis Paradis).
- Added the ability to read/write blocks using logical block data (lbdata) option. While this new feature was added specifically as a counterpart to the SCSI Write Same (disks) lbdata option, this option can be used with any device and/or data stream.

  What is Logical Block Data (lbdata) Format?

  This feature allows reading/writing of a 4-byte (32-bit) logical block address at the beginning of each data block tested. At the present time, the block number is stored using SCSI byte ordering (big-endian), which matchs what the SCSI Write Same command does.

  Several new 'dt' options have been added for this feature:

  ```
  lba=value       Set starting block used w/lbdata option.
  lbs=value       Set logical block size for lbdata option.
  [ ...as well as... ]
  enable=lbdata   Logical block data.    (Default: disabled)
  ```

  If this feature is enabled without specifying other options, the block size (lbs) defaults to 512, and the starting block address (lba) defaults to 0. Specifying "lba=" or "lbs=" options implicitly enables the "lbdata" feature.

  NOTES: The logical block number is *always* inserted starting at the beginning of each data block, not every "lbs" bytes (WRT to variable length opts). Also, the block number overwrites what the previous pattern bytes would have been (again matching the SCSI Write Same method). Also, enabling this feature will cause a degradation in performance statistics.

  If people would like an option to have the logical block stored in little-endian format, I'll consider adding an option *if* requested.

  Of course, the data verification routines have been modified to verify the logical block number matches the expected, and reports any mis-matches.
- For disk devices, the "min=" and "incr=" values now default to 512, when variable length transfers are requested (previously defaulted to 1 which was great for tapes, but not so good for block devices). [A future version of 'dt' will obtain the device block size via
  the new DEVGETINFO ioctl() now available in Platinum (ptos).]
- Removed "enable=ade" (Address Data Exception) test option. This was used to test kernel unaligned data exception fixups, but it's seldom (if ever) used, and it's time to cleanup code.

- Added "ttymin=" option for setting the VMIN value for serial line testing. Also fixed problem where VMIN got set incorrectly if the block size was greater than 255 (VMIN is a u_char on Tru64 Unix).

New Features in *dt* Version 8.0:

- On compare errors, data dumping now occurs by default. The "dlimit=value" option can be used to override the default dump limit of 64 bytes (use "disable=dump" to turn dumping off).
- On compare errors, the relative block number is reported for disk devices (see example below).
- The "align=rotate" option is now effective when selecting AIO testing (was being ignored before). Also corrected AIO end of media error handling (wasn't done consistently).
- Added "oncerr={abort|continue}" option to control child process errors. By default, 'dt' waits for all child processes to exit, regardless of whether an error is detected. When "oncerr=abort" is specified, 'dt' aborts all child prcoesses when _any_ child exits with an error status (effectively a halt on error).
- Added special mapping of pattern strings as shown below:

```
Pattern String Input:
    \\ = Backslash   \a = Alert (bell)   \b = Backspace
    \f = Formfeed    \n = Newline        \r = Carriage Return
    \t = Tab         \v = Vertical Tab   \e or \E = Escape
    \ddd = Octal Value    \xdd or \Xdd = Hexadecimal Value
```

- Minor changes were made to allow modem control testing to work. Normally, 'dt' disables modem control (aka "stty local"), but specifying "enable=modem" sets modem control (aka "stty -local"), On Tru64 UNIX systems, asserted modem signals get displayed at start of test, when debug is enabled.

  NOTE: Ensure the cable used for testing supports modem signals (DSR, CD, CTS, & RTS) or loss of data or partial reads may occur, especially at high speeds. The terminal driver may simply block read/write requests waiting for assertion of certain modem signals.
- The "Total Statistics" report has been enhanced to display the pattern string "pattern=string" or the pattern file size "pf=" when these options are used.
- This version of *dt*, was also ported to Sun's Solaris.

New Features in *dt* Version 7.0:

```
"procs="          To initiate multiple processes.
"files="          To process multiple tape files.
"step="           To force seeks after disk I/O.
"runtime="        To control how long to run.
"enable=aio"      To enable POSIX Asynchronous I/O.
"aios=value"      To set POSIX AIO's to queue (default 8).
"align=rotate"    To rotate data address through sizeof(ptr).
"pattern=incr"    To use an incrementing data pattern.
"min="            To set the minimum record size.
"max="            To Set the maximum record size.
"incr="           To set the record bytes to increment.
```

The "*procs=*" option is useful for load testing, and testing buffer cache, driver, and controller optimizations. The "*files=*" option is useful for verifying tape drivers properly report and reset file marks, end of recorded media, and end of tape conditions. The "*step=*" option is useful to cause non-sequential disk access, but "*iotype=random*" can also be used. The "*min=*", "*max=*", and "*incr=*" options are used for variable length records. In addition to these options, errors now have a timestamp associated with them, and the program start and end times are displayed in the total statistics.

There are also several useful scripts I've developed which use *dt* for testing. These script resides in the ~rmiller/utils/dt directory along with the source code and pattern files required for testing with the RRD TEST DISC. A short overview of these scripts is described below:

- dta - For testing asynchronous terminal lines.
- dtc - Test the RRD40/42 using the RRD TEST DISC.
- dtf - For testing floppy drives.
- dtt - For testing tape devices.
- dtr - To read data patterns written by 'dtw'.
- dtw - Writes the data pattens obtained from the RRD TEST DISC. These pattern files are named pattern_[0-9].

A common script could be developed to replace most of the existing scripts, but I haven't taken the time to do this yet. These scripts can be used as templates for developing specialized test scripts.

NOTE: The scripts are written to use the C-shell *csh*, but work just fine with the public domain *tcsh*. On my QNX system, I just created a link from *tcsh* to *csh* and the scripts ran without any problems.

% **dt help**

```
Usage: dt options...

Usage: dt options...

    Where options are:
        if=filename      The input file to read.
        of=filename      The output file to write.
        pf=filename      The data pattern file to use.
        bs=value         The block size to read/write.
        log=filename     The log file name to write.
        munsa=string     Set munsa to: cr, cw, pr, pw, ex.
        aios=value       Set number of AIO's to queue.
        align=offset     Set offset within page aligned buffer.
    or  align=rotate     Rotate data address through sizeof(ptr).
        dispose=mode     Set file dispose to: {delete or keep}.
        dlimit=value     Set the dump data buffer limit.
        dtype=string     Set the device type being tested.
        idtype=string    Set input device type being tested.
        odtype=string    Set output device type being tested.
        dsize=value      Set the device block (sector) size.
        errors=value     The number of errors to tolerate.
        files=value      Set number of tape files to process.
        flow=type        Set flow to: none, cts_rts, or xon_xoff.
        incr=value       Set number of record bytes to increment.
    or  incr=variable    Enables variable I/O request sizes.
```

```
    iodir=direction  Set I/O direction to: {forward or reverse}.
    iomode=mode      Set I/O mode to: {copy, test, or verify}.
    iotype=type      Set I/O type to: {random or sequential}.
    min=value        Set the minumum record size to transfer.
    max=value        Set the maximum record size to transfer.
    lba=value        Set starting block used w/lbdata option.
    lbs=value        Set logical block size for lbdata option.
    limit=value      The number of bytes to transfer.
    flags=flags      Set open flags:   {excl,sync,...}
    oflags=flags     Set output flags: {append,trunc,...}
    oncerr=action    Set child error action: {abort or continue}.
    parity=string    Set parity to: {even, odd, or none}.
    passes=value     The number of passes to perform.
    pattern=value    The 32 bit hex data pattern to use.
or  pattern=iot      Use DJ’s IOT test pattern.
or  pattern=incr     Use an incrementing data pattern.
or  pattern=string   The string to use for the data pattern.
    position=offset  Position to offset before testing.
    procs=value      The number of processes to create.
    ralign=value     The random I/O offset alignment.
    rlimit=value     The random I/O data byte limit.
    rseed=value      The random number generator seed.
    records=value    The number of records to process.
    runtime=time     The number of seconds to execute.
    slices=value     The number of disk slices to test.
    skip=value       The number of records to skip past.
    seek=value       The number of records to seek past.
    step=value       The number of bytes seeked after I/O.
    speed=value      The tty speed (baud rate) to use.
    timeout=value    The tty read timeout in .10 seconds.
    ttymin=value     The tty read minimum count (sets vmin).
    volumes=value    The number of volumes to process.
    vrecords=value   The record limit for the last volume.
    enable=flag      Enable one or more of the flags below.
    disable=flag     Disable one or more of the flags below.

Flags to enable/disable:
    aio              POSIX Asynchronous I/O.(Default: disabled)
    cerrors          Report close errors.   (Default: enabled)
    compare          Data comparison.       (Default: enabled)
    coredump         Core dump on errors.   (Default: disabled)
    debug            Debug output.          (Default: disabled)
    Debug            Verbose debug output.  (Default: disabled)
    rdebug           Random debug output.   (Default: disabled)
    diag             Log diagnostic msgs.   (Default: disabled)
    dump             Dump data buffer.      (Default: enabled)
    eei              Tape EEI reporting.    (Default: enabled)
    resets           Tape reset handling.   (Default: disabled)
```

```
        flush           Flush tty I/O queues.  (Default: enabled)
        fsync           Controls file sync'ing.(Default: runtime)
        header          Log file header.       (Default: enabled)
        lbdata          Logical block data.    (Default: disabled)
        loopback        Loopback mode.         (Default: disabled)
        microdelay      Microsecond delays.    (Default: disabled)
        mmap            Memory mapped I/O.     (Default: disabled)
        modem           Test modem tty lines.  (Default: disabled)
        multi           Multiple volumes.      (Default: disabled)
        pstats          Per pass statistics.   (Default: enabled)
        raw             Read after write.      (Default: disabled)
        stats           Display statistics.    (Default: enabled)
        table           Table(sysinfo) timing. (Default: disabled)
        ttyport         Flag device as a tty.  (Default: disabled)
        unique          Unique pattern.        (Default: enabled)
        verbose         Verbose output.        (Default: enabled)
        verify          Verify data written.   (Default: enabled)

        Example: enable=debug disable=compare,pstats

    MUNSA Lock Options:
        cr = Concurrent Read (permits read access, cr/pr/cw by others)
        pr = Protected Read (permits cr/pr read access to all, no write)
        cw = Concurrent Write (permits write and cr access to resource by all)
        pw = Protected Write (permits write access, cr by others)
        ex = Exclusive Mode (permits read/write access, no access to others)

            For more details, please refer to the dlm(4) reference page.

    Common Open Flags:
        excl (O_EXCL)         Exclusive open. (don't share)
        ndelay (O_NDELAY)     Non-delay open. (don't block)
        nonblock (O_NONBLOCK) Non-blocking open/read/write.
        rsync (O_RSYNC)       Synchronize read operations.
        sync (O_SYNC)         Sync updates for data/file attributes.

    Output Open Flags:
        append (O_APPEND)     Append data to end of existing file.
        defer (O_DEFER)       Defer updates to file during writes.
        dsync (O_DSYNC)       Sync data to disk during write operations.
        trunc (O_TRUNC)       Truncate an exisiting file before writing.

    Delays (Values are seconds, unless microdelay enabled):
        cdelay=value     Delay before closing the file.    (Def: 0)
        edelay=value     Delay between multiple passes.    (Def: 0)
        rdelay=value     Delay before reading each record. (Def: 0)
        sdelay=value     Delay before starting the test.   (Def: 0)
        tdelay=value     Delay before child terminates.    (Def: 1)
        wdelay=value     Delay before writing each record. (Def: 0)

    Numeric Input:
```

```
        For options accepting numeric input, the string may contain any
        combination of the following characters:

        Special Characters:
            w = words (4 bytes)            q = quadwords (8 bytes)
            b = blocks (512 bytes)         k = kilobytes (1024 bytes)
            m = megabytes (1048576 bytes)  p = page size (8192 bytes)
            g = gigabytes (1073741824 bytes)
            t = terabytes (1099511627776 bytes)
            inf or INF = infinity (18446744073709551615 bytes)

        Arithmetic Characters:
            + = addition                   - = subtraction
            * or x = multiplcation         / = division
            % = remainder

        Bitwise Characters:
            ~ = complement of value       >> = shift bits right
           << = shift bits left            & = bitwise 'and' operation
            | = bitwise 'or' operation     ^ = bitwise exclusive 'or'

        The default base for numeric input is decimal, but you can override
        this default by specifying 0x or 0X for hexadecimal conversions, or
        a leading zero '0' for octal conversions.  NOTE: Evaluation is from
        right to left without precedence, and parenthesis are not permitted.

    Pattern String Input:
            \\ = Backslash   \a = Alert (bell)   \b = Backspace
            \f = Formfeed    \n = Newline        \r = Carriage Return
            \t = Tab         \v = Vertical Tab   \e or \E = Escape
            \ddd = Octal Value    \xdd or \Xdd = Hexadecimal Value

    Time Input:
            d = days (86400 seconds),      h = hours (3600 seconds)
            m = minutes (60 seconds),      s = seconds (the default)

        Arithmetic characters are permitted, and implicit addition is
        performed on strings of the form '1d5h10m30s'.

    Defaults:
        errors=1, files=0, passes=1, records=0, bs=512, log=stderr
        pattern=0x39c39c39, flow=xon_xoff, parity=none, speed=9600
        timeout=3 seconds, dispose=delete, align=0 (page aligned)
        aios=8, dlimit=64, oncerr=continue, volumes=0, vrecords=1
        iodir=forward, iomode=test, iotype=sequential

    --> Date: February 1st, 2001, Version: 14.1, Author: Robin T. Miller <--
%
```

APPENDIX B
dt Examples

This section contains various *dt* examples used to show its' capabilities and to help get new users started. A short description prefaces each test to describe the nature of the test being performed. Several of the latter tests, are real life problems which were either uncovered directly by *dt*, or were easily reproduced using a specific *dt* command lines which helps trouble-shooting problems.

On Tru64 UNIX systems, next to the device name in the total statistics, you'll notice the device name and device type. This information is obtained by using the DEC specific DEVIOCGET I/O control command. This is very useful for identifying the device under test, especially since performance and various problems are device specific. For non-Tru64 UNIX systems you'll only see the device type displayed, not the real device name, which is setup based on known system dependent device naming conventions (e.g., "/dev/ser" prefix for QNX serial ports, "/dev/cd" or "/dev/scd" prefix for Linux CD-ROM devices).

TEST DESCRIPTION: This test does read testing of a raw disk partition with data comparisons disabled, using the POSIX Asynchronous I/O API (AIO is only supported on Tru64 UNIX at this time).

```
% dt if=/dev/rrz3c bs=8k limit=50m disable=compare enable=aio
Total Statistics:
      Input device/file name: /dev/rrz3c (Device: RZ25, type=disk)
           Data pattern read: 0x39c39c39 (data compare disabled)
     Total records processed: 6400 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 52428800 (51200.000 Kbytes, 50.000 Mbytes)
      Average transfer rates: 2227853 bytes/sec, 2175.637 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m23.53s
           Total system time: 00m01.36s
             Total user time: 00m00.20s
               Starting time: Wed Sep 15 12:47:55 1993
                 Ending time: Wed Sep 15 12:48:18 1993
%
```

TEST DESCRIPTION: This test does a write/read verify pass of a 50Mb file through the UFS file system, with the file disposition set to "keep", so the test file is not deleted. Normally, *dt* deletes the test files created before exiting.

% dt of=/usr/tmp/x bs=8k limit=50m dispose=keep

```
Write Statistics:
     Total records processed: 6400 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 52428800 (51200.000 Kbytes, 50.000 Mbytes)
      Average transfer rates: 1530768 bytes/sec, 1494.891 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 0/1
```

```
          Total elapsed time: 00m34.25s
           Total system time: 00m03.48s
             Total user time: 00m06.70s

Read Statistics:
     Total records processed: 6400 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 52428800 (51200.000 Kbytes, 50.000 Mbytes)
      Average transfer rates: 2243743 bytes/sec, 2191.155 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m23.36s
           Total system time: 00m02.05s
             Total user time: 00m13.95s

Total Statistics:
     Output device/file name: /usr/tmp/x
   Data pattern read/written: 0x39c39c39
     Total records processed: 12800 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 104857600 (102400.000 Kbytes, 100.000 Mbytes)
      Average transfer rates: 1819918 bytes/sec, 1777.264 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m57.61s
           Total system time: 00m05.55s
             Total user time: 00m20.65s
               Starting time: Wed Sep 15 13:42:05 1993
                 Ending time: Wed Sep 15 13:43:03 1993
% ls -ls /usr/tmp/x
51240 -rw-r--r--   1 rmiller  system   52428800 Sep 15 13:42 /usr/tmp/x
% od -x < /usr/tmp/x
0000000  9c39 39c3 9c39 39c3 9c39 39c3 9c39 39c3
*
310000000
%
```

TEST DESCRIPTION: This test does a read verify pass of the 50Mb file created in the previous test, using the memory mapped I/O API (mmap is only supported on SUN/OS and Tru64 UNIX at this time).

```
% dt if=/usr/tmp/x bs=8k limit=50m enable=mmap
Total Statistics:
      Input device/file name: /usr/tmp/x
           Data pattern read: 0x39c39c39
     Total records processed: 6400 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 52428800 (51200.000 Kbytes, 50.000 Mbytes)
      Average transfer rates: 2282821 bytes/sec, 2229.318 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
```

```
          Total elapsed time: 00m22.96s
           Total system time: 00m01.13s
             Total user time: 00m07.73s
               Starting time: Wed Sep 15 13:49:10 1993
                 Ending time: Wed Sep 15 13:49:33 1993
```

% **rm /usr/tmp/x**
%

TEST DESCRIPTION: This test does a write/read verify pass to a QIC-320 1/4" tape drive. Please notice the total average transfer rate. This lower rate is caused by the tape rewind performed after writing the tape. This rewind time is not included in the write/read times, but is part of the total time.

% **dt of=/dev/rmt0h bs=64k limit=10m**

```
Write Statistics:
     Total records processed: 160 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 10485760 (10240.000 Kbytes, 10.000 Mbytes)
      Average transfer rates: 157365 bytes/sec, 153.677 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 01m06.63s
           Total system time: 00m00.10s
             Total user time: 00m01.33s

Read Statistics:
     Total records processed: 160 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 10485760 (10240.000 Kbytes, 10.000 Mbytes)
      Average transfer rates: 194842 bytes/sec, 190.276 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m53.81s
           Total system time: 00m00.08s
             Total user time: 00m02.78s

Total Statistics:
     Output device/file name: /dev/rmt0h (Device: TZK10, type=tape)
   Data pattern read/written: 0x39c39c39
     Total records processed: 320 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 20971520 (20480.000 Kbytes, 20.000 Mbytes)
      Average transfer rates: 115950 bytes/sec, 113.233 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 03m00.86s
           Total system time: 00m00.18s
             Total user time: 00m04.11s
               Starting time: Wed Sep 15 11:50:36 1993
                 Ending time: Wed Sep 15 11:53:50 1993
%
```

TEST DESCRIPTION: This test does a write/read verify pass of 2 tape files to a DEC TZ86 tape drive using variable length records ranging from 10 Kbytes to 100 Kbytes using the default increment value of 1 byte.

```
% dt of=/dev/rmt1h min=10k max=100k limit=5m files=2
Write Statistics:
     Total records processed: 1000 with min=10240, max=102400, incr=1
     Total bytes transferred: 10485760 (10240.000 Kbytes, 10.000 Mbytes)
      Average transfer rates: 642641 bytes/sec, 627.579 Kbytes/sec
      Total passes completed: 0/1
       Total files processed: 2/2
       Total errors detected: 0/1
          Total elapsed time: 00m16.31s
           Total system time: 00m00.28s
             Total user time: 00m01.26s

Read Statistics:
     Total records processed: 1000 with min=10240, max=102400, incr=1
     Total bytes transferred: 10485760 (10240.000 Kbytes, 10.000 Mbytes)
      Average transfer rates: 214725 bytes/sec, 209.693 Kbytes/sec
      Total passes completed: 1/1
       Total files processed: 2/2
       Total errors detected: 0/1
          Total elapsed time: 00m48.83s
           Total system time: 00m00.45s
             Total user time: 00m30.95s

Total Statistics:
     Output device/file name: /dev/rmt1h (Device: TZ86, type=tape)
   Data pattern read/written: 0x39c39c39
     Total records processed: 2000 with min=10240, max=102400, incr=1
     Total bytes transferred: 20971520 (20480.000 Kbytes, 20.000 Mbytes)
      Average transfer rates: 229322 bytes/sec, 223.948 Kbytes/sec
      Total passes completed: 1/1
       Total files processed: 4/4
       Total errors detected: 0/1
          Total elapsed time: 01m31.45s
           Total system time: 00m00.75s
             Total user time: 00m32.21s
               Starting time: Mon Sep 13 15:29:23 1993
                 Ending time: Mon Sep 13 15:31:00 1993
%
```

TEST DESCRIPTION: This test does writing/reading through a pipe. Notice the special character '-' which indicates write standard out/read standard in.

```
% dt of=- bs=8k limit=1g disable=stats | dt if=- bs=8k limit=1g
Total Statistics:
```

```
      Input device/file name: -
           Data pattern read: 0x39c39c39
     Total records processed: 131072 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
      Average transfer rates: 2334644 bytes/sec, 2279.926 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 07m39.91s
           Total system time: 00m17.65s
             Total user time: 04m44.66s
               Starting time: Wed Sep 15 11:40:08 1993
                 Ending time: Wed Sep 15 11:47:48 1993
%
```

TEST DESCRIPTION: This test does writing/reading through a fifo (named pipe). This is similar to the previous test, except a fifo file is created, and a single invocation of *dt* is used for testing.

```
% mkfifo NamedPipe
% ls -ls NamedPipe
0 prw-r--r--   1 rmiller  system         0 Sep 16 09:52 NamedPipe
% dt of=NamedPipe bs=8k limit=1g enable=loopback
Total Statistics:
     Output device/file name: NamedPipe (device type=fifo)
        Data pattern written: 0x39c39c39 (read verify disabled)
     Total records processed: 131072 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
      Average transfer rates: 2264402 bytes/sec, 2211.330 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 07m54.18s
           Total system time: 00m21.80s
             Total user time: 02m14.96s
               Starting time: Thu Sep 16 09:42:24 1993
                 Ending time: Thu Sep 16 09:50:18 1993

Total Statistics:
      Input device/file name: NamedPipe (device type=fifo)
           Data pattern read: 0x39c39c39
     Total records processed: 131072 @ 8192 bytes/record (8.000 Kbytes)
     Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
      Average transfer rates: 2264402 bytes/sec, 2211.330 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 07m54.18s
           Total system time: 00m19.90s
             Total user time: 04m44.01s
               Starting time: Thu Sep 16 09:42:24 1993
                 Ending time: Thu Sep 16 09:50:19 1993
```

% **rm NamedPipe**
%

TEST DESCRIPTION: This test performs a loopback test between two serial lines. Debug was enabled to display additional test information, which is useful if serial line testing hangs. *dt* does not use a watchdog timer.

Also notice the number of bytes allocated was 68, not 64 as '*bs=*' indicates. Pad bytes are allocated at the end of data buffers and checked after reads to ensure drivers/file system code do not write too many bytes (this has uncovered DMA FIFO flush problems in device drivers in the past).

% **dt if=/dev/tty00 of=/dev/tty01 bs=64 limit=100k flow=xon_xoff parity=none speed=38400 enable=debug**

```
dt: Attempting to open input file '/dev/tty00', mode = 00...
dt: Input file '/dev/tty00' successfully opened, fd = 3
dt: Saving current terminal characteristics, fd = 3...
dt: Setting up test terminal characteristics, fd = 3...
dt: Attempting to open output file '/dev/tty01', mode = 01...
dt: Output file '/dev/tty01' successfully opened, fd = 4
dt: Saving current terminal characteristics, fd = 4...
dt: Setting up test terminal characteristics, fd = 4...
dt: Parent PID = 1809, Child PID = 1810
dt: Allocated buffer at address 0x4a000 of 68 bytes, using offset 0
dt: Allocated buffer at address 0x4a000 of 68 bytes, using offset 0
dt: Characters remaining in output queue = 304
dt: Waiting for output queue to drain...
dt: Output queue finished draining...
Total Statistics:
     Output device/file name: /dev/tty01 (device type=terminal)
    Terminal characteristics: flow=xon_xoff, parity=none, speed=38400
        Data pattern written: 0x39c39c39 (read verify disabled)
     Total records processed: 1600 @ 64 bytes/record (0.063 Kbytes)
     Total bytes transferred: 102400 (100.000 Kbytes, 0.098 Mbytes)
      Average transfer rates: 3840 bytes/sec, 3.750 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m26.66s
           Total system time: 00m00.06s
             Total user time: 00m00.01s
               Starting time: Wed Sep 15 11:37:39 1993
                 Ending time: Wed Sep 15 11:38:07 1993

Total Statistics:
      Input device/file name: /dev/tty00 (device type=terminal)
    Terminal characteristics: flow=xon_xoff, parity=none, speed=38400
           Data pattern read: 0x39c39c39
     Total records processed: 1600 @ 64 bytes/record (0.063 Kbytes)
     Total bytes transferred: 102400 (100.000 Kbytes, 0.098 Mbytes)
      Average transfer rates: 3703 bytes/sec, 3.617 Kbytes/sec
      Total passes completed: 1/1
```

```
       Total errors detected: 0/1
          Total elapsed time: 00m27.65s
           Total system time: 00m00.28s
             Total user time: 00m00.05s
               Starting time: Wed Sep 15 11:37:39 1993
                 Ending time: Wed Sep 15 11:38:07 1993

dt: Restoring saved terminal characteristics, fd = 3...
dt: Closing file '/dev/tty00', fd = 3...
dt: Waiting for child PID 1810 to exit...
dt: Child PID 1810, exited with status = 0
dt: Restoring saved terminal characteristics, fd = 4...
dt: Closing file '/dev/tty01', fd = 4...
%
```

TEST DESCRIPTION: This test does write/read testing to a raw device starting 2 processes, each of which will execute 2 passes. Notice the IOT pattern is specified, to avoid possible data compare failures. Normally a different pattern gets used for each pass. There are 12 different patterns which get cycled through, *if* a data pattern was **not** specified on the command line. Since each process runs at an indeterminate speed, it's possible for one process to be writing a different pattern, while the other process is still reading the previous pattern, which results in false data comparison failures. Please beware of this, until this issue is resolved in a future release.

% **dt of=/dev/rrz2c bs=64k limit=1g pattern=iot procs=2**

```
Write Statistics (29090):
    Current Process Reported: 1/2
     Total records processed: 16384 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
      Average transfer rates: 4176629 bytes/sec, 4078.740 Kbytes/sec
     Number I/O's per second: 63.730
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 04m17.08s
           Total system time: 00m03.43s
             Total user time: 00m31.96s

Write Statistics (29105):
    Current Process Reported: 2/2
     Total records processed: 16384 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
      Average transfer rates: 4175005 bytes/sec, 4077.154 Kbytes/sec
     Number I/O's per second: 63.706
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 04m17.18s
           Total system time: 00m02.81s
             Total user time: 00m32.35s
    •
```

```
    •
    •
Total Statistics (29090):
     Output device/file name: /dev/rrz2c (Device: BB01811C, type=disk)
     Type of I/O's performed: sequential
    Current Process Reported: 1/2
    Data pattern string used: 'IOT Pattern'
     Total records processed: 32768 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 2147483648 (2097152.000 Kbytes, 2048.000 Mbytes)
      Average transfer rates: 3615597 bytes/sec, 3530.856 Kbytes/sec
     Number I/O's per second: 55.170
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 09m53.95s
           Total system time: 00m06.63s
             Total user time: 02m32.51s
               Starting time: Thu Nov  9 09:50:28 2000
                 Ending time: Thu Nov  9 10:00:22 2000

Total Statistics (29105):
     Output device/file name: /dev/rrz2c (Device: BB01811C, type=disk)
     Type of I/O's performed: sequential
    Current Process Reported: 2/2
    Data pattern string used: 'IOT Pattern'
     Total records processed: 32768 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 2147483648 (2097152.000 Kbytes, 2048.000 Mbytes)
      Average transfer rates: 3604370 bytes/sec, 3519.893 Kbytes/sec
     Number I/O's per second: 54.998
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 09m55.80s
           Total system time: 00m05.90s
             Total user time: 02m38.21s
               Starting time: Thu Nov  9 09:50:28 2000
                 Ending time: Thu Nov  9 10:00:24 2000
%
```

TEST DESCRIPTION: This test attempts to write to a Tru64 UNIX raw disk which contains a valid label block, and the action necessary to destroy this label block before writes are possible. As you can see, the first disk block (block 0) is write protected (all other blocks are not however). Since many people, including myself, have been burnt (mis-lead) by this wonderful feature, I thought it was worth documenting here.

```
# file /dev/rrz11c
/dev/rrz11c: character special (8/19458) SCSI #1 RZ56 disk #88 (SCSI ID #3)
# disklabel -r -w /dev/rrz11c rz56
# ls -ls /dev/rrz11a
0 crw-rw-rw-   1 root     system     8,19456 Sep 15 11:33 /dev/rrz11a
```

```
# dt of=/dev/rrz11c bs=64k limit=1m disable=stats
dt: 'write' - Read-only file system
dt: Error number 1 occurred on Thu Sep 16 10:53:54 1993
# dt of=/dev/rrz11a position=1b bs=64k limit=1m disable=stats
# disklabel -z /dev/rrz11c
# dt of=/dev/rrz11c bs=64k limit=1m disable=stats
#
```

TEST DESCRIPTION: This test shows a real life problem discovered on a DEC 3000-500 (Flamingo) system using Tru64 UNIX V1.3. This test uncovers a data corruption problem that occurs at the end of data buffers on read requests. The problem results from the FIFO being improperly flushed when DMA transfers abort on certain boundaries (residual bytes left in FIFO). This failure is uncovered by performing large reads of short records and verifying the pad bytes, allocated at the end of data buffers, do **not** get inadvertantly overwritten.

```
% dt of=/dev/rmt0h min=1k max=25k incr=p-1 limit=1m disable=stats,verify
% dt if=/dev/rmt0h min=1k+25 max=25k incr=p-1 limit=1m disable=stats
dt: WARNING: Record #1, attempted to read 1049 bytes, read only 1024 bytes.
dt: WARNING: Record #2, attempted to read 9240 bytes, read only 9215 bytes.
dt: Data compare error at pad byte 0 in record number 2
dt: Data expected = 0xc6, data found = 0xff
dt: Error number 1 occurred on Sat Sep 18 11:15:08 1993
% dt if=/dev/rmt0h min=1k+25 max=25k incr=p-1 limit=1m disable=stats
dt: WARNING: Record #1, attempted to read 1049 bytes, read only 1024 bytes.
dt: WARNING: Record #2, attempted to read 9240 bytes, read only 9215 bytes.
dt: Data compare error at pad byte 0 in record number 2
dt: Data expected = 0xc6, data found = 0xff
dt: Data buffer pointer = 0x3e3ff, buffer offset = 9215

Dumping Buffer (base = 0x3c000, offset = 9215, size = 9219 bytes):

0x3e3df  39 39 9c c3 39 39 9c c3 39 39 9c c3 39 39 9c c3
0x3e3ef  39 39 9c c3 39 39 9c c3 39 39 9c c3 39 39 9c c3
0x3e3ff  ff c6 63 3c c6 c6 63 3c c6 c6 63 3c c6 c6 63 3c
0x3e40f  c6 c6 63 3c c6 c6 63 3c c6 c6 63 3c c6 c6 63 3c
dt: Error number 1 occurred on Sat Sep 18 11:15:31 1993
% echo $status
-1
%
```

TEST DESCRIPTION: This test shows a real life problem discovered on a DEC 7000 (Ruby) system using Tru64 UNIX V1.3. A simple variable length record test is performed, and as you can see, reading the same record size written runs successfully. The failure does **not** occur until large reads of the short tape records previously written is performed.

Upon reviewing this problem on an SDS-3F SCSI analyzer, it appears this device does a disconnect/save data pointers/reconnect sequence, followed by the check condition status, which is not being handled properly by someone (either our CAM *xza* driver, or the KZMSA firmware... this is still being investigated). The problem results in wrong record sizes being returned, and in this example, the first record is returned with a count of zero which looks like an end of file indication. The *tapex -g* option "*Random record-size tests*" originally found this problem, but as you can see, *dt* was able to easily reproduce this problem.

% dt of=/dev/rmt12h min=2k+10 max=250k incr=p-3 records=10

```
Write Statistics:
     Total records processed: 10 with min=2058, max=256000, incr=8189
     Total bytes transferred: 389085 (379.966 Kbytes, 0.371 Mbytes)
      Average transfer rates: 2593900 bytes/sec, 2533.105 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 00m00.15s
           Total system time: 00m00.00s
             Total user time: 00m00.03s

Read Statistics:
     Total records processed: 10 with min=2058, max=256000, incr=8189
     Total bytes transferred: 389085 (379.966 Kbytes, 0.371 Mbytes)
      Average transfer rates: 188267 bytes/sec, 183.854 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m02.06s
           Total system time: 00m00.01s
             Total user time: 00m00.63s

Total Statistics:
     Output device/file name: /dev/rmt12h (Device: TZ86, type=tape)
   Data pattern read/written: 0x39c39c39
     Total records processed: 20 with min=2058, max=256000, incr=8189
     Total bytes transferred: 778170 (759.932 Kbytes, 0.742 Mbytes)
      Average transfer rates: 53239 bytes/sec, 51.991 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m14.61s
           Total system time: 00m00.01s
             Total user time: 00m00.66s
               Starting time: Sat Sep 18 12:33:26 1993
                 Ending time: Sat Sep 18 12:34:44 1993
```

% dt if=/dev/rmt12h bs=250k records=10

```
Total Statistics:
      Input device/file name: /dev/rmt12h (Device: TZ86, type=tape)
           Data pattern read: 0x39c39c39
     Total records processed: 0 @ 256000 bytes/record (250.000 Kbytes)
     Total bytes transferred: 0 (0.000 Kbytes, 0.000 Mbytes)
      Average transfer rates: 0 bytes/sec, 0.000 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m01.31s
           Total system time: 00m00.00s
             Total user time: 00m00.01s
               Starting time: Sat Sep 18 12:40:15 1993
                 Ending time: Sat Sep 18 12:40:22 1993
```

% **dt if=/dev/rmt12h bs=250k records=10 enable=debug**

```
dt: Attempting to open input file '/dev/rmt12h', mode = 00...
dt: Input file '/dev/rmt12h' successfully opened, fd = 3
dt: Allocated buffer at address 0x52000 of 256004 bytes, using offset 0
dt: End of file/tape/media detected, count = 0, errno = 0
dt: Exiting with status code 254...
Total Statistics:
      Input device/file name: /dev/rmt12h (Device: TZ86, type=tape)
           Data pattern read: 0x39c39c39
     Total records processed: 0 @ 256000 bytes/record (250.000 Kbytes)
     Total bytes transferred: 0 (0.000 Kbytes, 0.000 Mbytes)
      Average transfer rates: 0 bytes/sec, 0.000 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m01.30s
           Total system time: 00m00.01s
             Total user time: 00m00.01s
               Starting time: Sat Sep 18 12:40:36 1993
                 Ending time: Sat Sep 18 12:40:43 1993

dt: Closing file '/dev/rmt12h', fd = 3...
%
```

TEST DESCRIPTION: This test shows a real life problem discovered on a DEC 3000-500 (Flamingo) system using Tru64 UNIX V1.3. The test uncovers a problem issuing I/O requests with large transfer sizes (>2.5 megabytes). I don't know the specifics of correcting this problem, which is not important in this context, but the failure indication was that *dt* never completed (the process appeared hung... actually sleeping waiting for I/O completion).

When a failure like this occurs, it is oftentimes useful to see where in the kernel the process is sleeping. This example shows how to identify the *dt* process ID (PID), and how to use *dbx* to map that process and obtain a kernel stack traceback. This seems like useful debugging information to include here, since my experience is that many people are unaware of how to trouble-shoot these types of problems.

% **file /dev/rrz11c**

```
/dev/rrz11c: character special (8/19458) SCSI #1 RZ56 disk #88 (SCSI ID #3)
% dt if=/dev/rrz11a bs=3m count=1 enable=debug
dt: Attempting to open input file '/dev/rrz11a', mode = 00...
dt: Input file '/dev/rrz11a' successfully opened, fd = 3
dt: Allocated buffer at address 0x52000 of 3145732 bytes, using offset 0
[ Ctrl/Z typed to background hung dt process at this point. ]
Stopped
% ps ax | fgrep dt
 4512 p6  U      0:00.48 dt if=/dev/rrz11a bs=3m count=1 enable=debug
 4514 p6  S      0:00.02 fgrep dt
% dbx -k /vmunix /dev/mem
dbx version 3.11.1
Type 'help' for help.

stopped at
warning: Files compiled -g3: parameter values probably wrong
  [thread_block:1414 ,0xfffffc00002ddf80]  Source not available
(dbx) set $pid=4512
stopped at  [thread_block:1414 ,0xfffffc00002ddf80]  Source not available
(dbx) trace
>  0 thread_block() ["../../../../src/kernel/kern/sched_prim.c":1414, 0xfffffc00002ddf80]
   1 mpsleep(chan = 0xffffffff8445fea0 = "...", pri = 0x18, wmesg = 0xfffffc000042a258 = "event", timo = 0x0, lockp = (nil), flags = 0x1) ["../../../../src/kernel/bsd/kern_synch.c":278, 0xfffffc0000264934]
   2 event_wait(evp = 0x52000, interruptible = 0x0, timo = 0x0) ["../../../../src/kernel/kern/event.c":137, 0xfffffc00002ce2a8]
   3 biowait(bp = 0xffffffff8434ba00) ["../../../../src/kernel/vfs/vfs_bio.c":904, 0xfffffc00002a458c]
   4 physio(strat = 0xfffffc00003956d0, bp = 0x14002aac0, dev = 0x804c00, rw = 0x1, mincnt = 0xfffffc00003a4af0, uio = 0xffffffff8434ba00) ["../../../../src/kernel/ufs/ufs_physio.c":205, 0xfffffc0000291fbc]
   5 cdisk_read(dev = 0x804c00, uio = 0xffffffff8d5f1d68) ["../io/cam/cam_disk.c":2249, 0xfffffc0000396d5c]
   6 spec_read(vp = 0x14c, uio = 0xffffffff84474640, ioflag = 0x0, cred = 0xffffffff843e9360) ["../../../../src/kernel/vfs/spec_vnops.c":1197, 0xfffffc00002a2614]
   7 ufsspec_read(vp = (nil), uio = 0xffffffff843e9360, ioflag = 0xffffffff84342030, cred = 0x352000) ["../../../../src/kernel/ufs/ufs_vnops.c":2731, 0xfffffc0000298e68]
   8 vn_read(fp = 0xffffffff8c74f428, uio = 0xffffffff8d5f1d68, cred = 0xffffffff843e9360) ["../../../../src/kernel/vfs/vfs_vnops.c":580, 0xfffffc00002ae3c8]
   9 rwuio(p = 0xffffffff8c763490, fdes = 0xffffffff844f5cc0, uio = 0xffffffff8d5f1d68, rw = UIO_READ, retval = 0xffffffff8d5f1e40) ["../../../../src/kernel/bsd/sys_generic.c":351, 0xfffffc000026ce34]
  10 read(p = 0xffffffff8d5f1d58, args = 0xffffffff00000001, retval = (nil)) ["../../../../src/kernel/bsd/sys_generic.c":201, 0xfffffc000026caf0]
  11 syscall(ep = 0xffffffff8d5f1ef8, code = 0x3) ["../../../../src/kernel/arch/alpha/syscall_trap.c":593, 0xfffffc0000379698]
  12 _Xsyscall() ["../../../../src/kernel/arch/alpha/locore.s":751, 0xfffffc000036c550]
(dbx) quit
%
```

TEST DESCRIPTION: This test shows a real life problem discovered on a DEC 3000-500 (Flamingo) system using Tru64 UNIX V3.2. The test uncovers a problem of too much data being copied to the user buffer, when long reads of short tape records are performed.

```
% dt of=/dev/rmt0h min=10k max=64k count=100
Write Statistics:
     Total records processed: 100 with min=10240, max=65536, incr=1
     Total bytes transferred: 1028950 (1004.834 Kbytes, 0.981 Mbytes)
      Average transfer rates: 61552 bytes/sec, 60.110 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 00m16.71s
```

```
           Total system time: 00m00.03s
             Total user time: 00m00.23s

Read Statistics:
     Total records processed: 100 with min=10240, max=65536, incr=1
     Total bytes transferred: 1028950 (1004.834 Kbytes, 0.981 Mbytes)
      Average transfer rates: 150946 bytes/sec, 147.408 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m06.81s
           Total system time: 00m00.05s
             Total user time: 00m00.48s

Total Statistics:
     Output device/file name: /dev/rmt0h (Device: TZK10, type=tape)
   Data pattern read/written: 0x39c39c39
     Total records processed: 200 with min=10240, max=65536, incr=1
     Total bytes transferred: 2057900 (2009.668 Kbytes, 1.963 Mbytes)
      Average transfer rates: 53966 bytes/sec, 52.701 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m38.13s
           Total system time: 00m00.08s
             Total user time: 00m00.71s
               Starting time: Tue Nov 14 16:06:54 1995
                 Ending time: Tue Nov 14 16:07:37 1995

im2fast% dt if=/dev/rmt0h min=20k max=64k count=100
dt: WARNING: Record #1, attempted to read 20480 bytes, read only 10240 bytes.
dt: WARNING: Record #2, attempted to read 20481 bytes, read only 10241 bytes.
dt: Data compare error at inverted byte 10242 in record number 2
dt: Data expected = 0x63, data found = 0xff, pattern = 0xc63c63c6
dt: The incorrect data starts at address 0x140012801 (marked by asterisk ’*’)
dt: Dumping Data Buffer (base = 0x140010000, offset = 10241, limit = 64 bytes):

0x1400127e1  9c c3 39 39 9c c3 39 39 9c c3 39 39 9c c3 39 39
0x1400127f1  9c c3 39 39 9c c3 39 39 9c c3 39 39 9c c3 39 39
0x140012801 *ff ff 03 c6 63 3c c6 c6 63 3c c6 c6 63 3c c6 c6
0x140012811  63 3c c6 c6 63 3c c6 c6 63 3c c6 c6 63 3c c6 c6

dt: Error number 1 occurred on Tue Nov 14 17:54:28 1995
Total Statistics:
      Input device/file name: /dev/rmt0h (Device: TZK10, type=tape)
           Data pattern read: 0x39c39c39
     Total records processed: 2 with min=20480, max=65536, incr=1
     Total bytes transferred: 20481 (20.001 Kbytes, 0.020 Mbytes)
      Average transfer rates: 17810 bytes/sec, 17.392 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 1/1
          Total elapsed time: 00m01.15s
```

```
           Total system time: 00m00.01s
             Total user time: 00m00.01s
               Starting time: Tue Nov 14 17:54:22 1995
                 Ending time: Tue Nov 14 17:54:28 1995

im2fast%
```

DESCRIPTION: This example shows copying a partition with the bad block to another disk. I've used this operation to save my system disk more than once. This copy w/verify is also very useful for floppy diskettes which tend to be unreliable in my experience.

Note the use of "*errors=10*" so *dt* will continue after reading the bad block. Without this option *dt* exits after 1 error.

If copying an active file system, like your system disk, expect a couple verification errors since certain system files will likely get written. Whenever possible, the copy operation should be done on unmounted disks.

% dt if=/dev/rrz0b of=/dev/rrz3b iomode=copy errors=10 limit=5m

```
dt: 'read' - I/O error
dt: Relative block number where the error occcured is 428
dt: Error number 1 occurred on Fri Mar  7 10:53:15 1997
Copy Statistics:
    Data operation performed: Copied '/dev/rrz0b' to '/dev/rrz3b'.
     Total records processed: 20478 @ 512 bytes/record (0.500 Kbytes)
     Total bytes transferred: 10484736 (10239.000 Kbytes, 9.999 Mbytes)
      Average transfer rates: 87677 bytes/sec, 85.622 Kbytes/sec
      Total passes completed: 0/1
       Total errors detected: 1/10
          Total elapsed time: 01m59.58s
           Total system time: 00m06.26s
             Total user time: 00m00.35s

dt: 'read' - I/O error
dt: Relative block number where the error occcured is 428
dt: Error number 1 occurred on Fri Mar  7 10:55:11 1997
Verify Statistics:
    Data operation performed: Verified '/dev/rrz0b' with '/dev/rrz3b'.
     Total records processed: 20478 @ 512 bytes/record (0.500 Kbytes)
     Total bytes transferred: 10484736 (10239.000 Kbytes, 9.999 Mbytes)
      Average transfer rates: 359477 bytes/sec, 351.051 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 1/10
          Total elapsed time: 00m29.16s
           Total system time: 00m06.68s
             Total user time: 00m01.76s

Total Statistics:
      Input device/file name: /dev/rrz0b (Device: RZ28, type=disk)
     Total records processed: 40956 @ 512 bytes/record (0.500 Kbytes)
```

```
     Total bytes transferred: 20969472 (20478.000 Kbytes, 19.998 Mbytes)
      Average transfer rates: 140940 bytes/sec, 137.636 Kbytes/sec
      Total passes completed: 1/1
       Total errors detected: 2/10
          Total elapsed time: 02m28.78s
           Total system time: 00m12.96s
             Total user time: 00m02.13s
               Starting time: Fri Mar  7 10:53:06 1997
                 Ending time: Fri Mar  7 10:55:35 1997

%
```

TEST DESCRIPTION: Here’s an example which shows the Extended Error Information (EEI) available for SCSI tapes on Tru64 Unix systems.

$ **dt if=/dev/rmt0h bs=16k limit=10m disable=compare**

```
dt: 'read' - I/O error

DEVIOGET ELEMENT        CONTENTS
----------------        --------
category                DEV_TAPE
bus                     DEV_SCSI
interface               SCSI
device                  TZK10
adpt_num                0
nexus_num               0
bus_num                 0
ctlr_num                0
slave_num               5
dev_name                tz
unit_num                5
soft_count              0
hard_count              16
stat                    0x108
                        DEV_WRTLCK DEV_HARDERR
category_stat           0x8000
                        DEV_10000_BPI

DEVGETINFO ELEMENT      CONTENTS
------------------      --------
media_status            0x10108
                        WrtProt HardERR POS_VALID
unit_status             0x131
                        Ready 1_FM_Close Rewind Buffered
record_size             512
density (current)       10000 BPI
density (on write)      16000 BPI
Filemark Cnt            0
Record Cnt              673
```

```
Class                   4 - QIC

MTIOCGET ELEMENT        CONTENTS
----------------        --------
mt_type                 MT_ISSCSI
mt_dsreg                0x108
                        DEV_WRTLCK DEV_HARDERR
mt_erreg                0x3 Nonrecoverable medium error.
mt_resid                31
mt_fileno               0
mt_blkno                673
DEV_EEI_STATUS
        version         0x1
        status          0x15  Device hardware error (hard error)
        flags           0x1000007
                        CAM_STATUS SCSI_STATUS SCSI_SENSE CAM_DATA
        cam_status      0x4  CCB request completed with an err
        scsi_status     0x2  SCSI_STAT_CHECK_CONDITION
        scsi_sense_data
                     Error Code: 0x70 (Current Error)
                      Valid Bit: 0x1 (Information field is valid)
                 Segment Number: 0
                      Sense Key: 0x3 (MEDIUM ERROR - Nonrecoverable medium error)
                 Illegal Length: 0
                   End Of Media: 0
                      File Mark: 0
              Information Field: 0x1f (31)
        Additional Sense Length: 0x16
   Command Specific Information: 0
Additional Sense Code/Qualifier: (0x3a, 0) = Medium not present
    Field Replaceable Unit Code: 0
           Sense Specific Bytes: 00 00 00
         Additional Sense Bytes: 00 02 a1 00 00 00 00 00 00 00 00 04

dt: Error number 1 occurred on Sat Sep 13 16:48:40 1997
Total Statistics:
      Input device/file name: /dev/rmt0h (Device: TZK10, type=tape)
           Data pattern read: 0x39c39c39 (data compare disabled)
     Total records processed: 21 @ 16384 bytes/record (16.000 Kbytes)
     Total bytes transferred: 344064 (336.000 Kbytes, 0.328 Mbytes)
      Average transfer rates: 70941 bytes/sec, 69.278 Kbytes/sec
     Number I/O’s per second: 4.330
      Total passes completed: 0/1
       Total errors detected: 1/1
          Total elapsed time: 00m04.85s
           Total system time: 00m00.01s
             Total user time: 00m00.00s
               Starting time: Sat Sep 13 16:48:26 1997
                 Ending time: Sat Sep 13 16:48:40 1997
```

```
$
```

DESCRIPTION: This example show a multiple volume test to a tape drive. The test ensures End Of Media (EOM) is detected properly, and that the close operation succeeds which indicated all buffered data and filemarks were written properly. Note: A short 7mm DAT tape was used for this test.

```
linux% dt of=/dev/st0 bs=32k files=4 limit=50m pattern=iot enable=multi
Please insert volume #2 in drive /dev/st0... Press ENTER when ready to proceed:
        [ Continuing in file #3, record #1425, bytes written so far 151519232... ]

Write Statistics:
     Total records processed: 6400 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 209715200 (204800.000 Kbytes, 200.000 Mbytes)
      Average transfer rates: 510529 bytes/sec, 498.564 Kbytes/sec
     Number I/O's per second: 15.580
      Total passes completed: 0/1
       Total files processed: 4/4
       Total errors detected: 0/1
          Total elapsed time: 06m50.78s
           Total system time: 00m00.75s
             Total user time: 00m06.90s

Please insert volume #1 in drive /dev/st0... Press ENTER when ready to proceed:
Please insert volume #2 in drive /dev/st0... Press ENTER when ready to proceed:
    [ Continuing in file #3, record #1425, bytes read so far 151519232... ]

Read Statistics:
     Total records processed: 6400 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 209715200 (204800.000 Kbytes, 200.000 Mbytes)
      Average transfer rates: 489657 bytes/sec, 478.181 Kbytes/sec
     Number I/O's per second: 14.943
      Total passes completed: 1/1
       Total files processed: 4/4
       Total errors detected: 0/1
          Total elapsed time: 07m08.29s
           Total system time: 00m00.91s
             Total user time: 00m26.53s

Total Statistics:
     Output device/file name: /dev/st0 (device type=tape)
     Type of I/O's performed: sequential
    Data pattern string used: 'IOT Pattern'
     Total records processed: 12800 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 419430400 (409600.000 Kbytes, 400.000 Mbytes)
      Average transfer rates: 434589 bytes/sec, 424.403 Kbytes/sec
     Number I/O's per second: 13.263
      Total passes completed: 1/1
       Total files processed: 8/8
       Total errors detected: 0/1
```

```
          Total elapsed time: 16m05.12s
           Total system time: 00m01.66s
             Total user time: 00m33.43s
               Starting time: Fri Feb 18 18:48:22 2000
                 Ending time: Fri Feb 18 19:04:28 2000

linux%
```

DESCRIPTION: This example shows the results of doing a read-after-write test to a floppy diskette. This option is valid with disks and tapes.

```
tru64% dt of=/dev/rfd0c min=b max=64k incr=7b iotype=random enable=raw runtime=3m
Read After Write Statistics:
     Total records processed: 100 with min=512, max=65536, incr=3584
     Total bytes transferred: 2949120 (2880.000 Kbytes, 2.812 Mbytes)
      Average transfer rates: 16923 bytes/sec, 16.526 Kbytes/sec
     Number I/O's per second: 0.574
      Total passes completed: 1
       Total errors detected: 0/1
          Total elapsed time: 02m54.26s
           Total system time: 00m00.01s
             Total user time: 00m00.16s

Total Statistics:
     Output device/file name: /dev/rfd0c (Device: floppy, type=disk)
     Type of I/O's performed: random (seed 0xa775f81)
   Data pattern read/written: 0x00ff00ff
     Total records processed: 109 with min=512, max=65536, incr=3584
     Total bytes transferred: 3011072 (2940.500 Kbytes, 2.872 Mbytes)
      Average transfer rates: 16642 bytes/sec, 16.252 Kbytes/sec
     Number I/O's per second: 0.602
      Total passes completed: 1
       Total errors detected: 0/1
          Total elapsed time: 03m00.93s
           Total system time: 00m00.01s
             Total user time: 00m00.16s
               Starting time: Wed Jan 12 16:38:38 2000
                 Ending time: Wed Jan 12 16:41:43 2000

tru64%
```

The following test shows starting 12 slices using the first 12 GBytes of disk space, writing/reading 1 MByte in each slice with the lbdata option enabled, and doing a read-after-write operation. The debug option is enabled simply to show the range of block for each slice.

```
tru64% dt of=/dev/rrz1c bs=256k capacity=12g limit=1m slices=12 enable=debug,lbdata,raw
dt: Data limit set to 12884901888 bytes (12288.000 Mbytes), 25165824 blocks.
dt: Started process 18122...
dt: Started process 18121...
```

```
dt: Started process 18127...
dt: Started process 18126...
dt: Started process 18128...
dt: Started process 18115...
dt: Started process 18133...
dt: Started process 18134...
dt: Started process 18132...
dt: Started process 18131...
dt (18122): Start Position 0 (lba 0), Limit 1048576, Pattern 0x39c39c39
dt (18122): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18121): Start Position 1073741824 (lba 2097152), Limit 1048576 bytes
dt (18121): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18127): Start Position 2147483648 (lba 4194304), Limit 1048576 bytes
dt (18127): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18126): Start Position 3221225472 (lba 6291456), Limit 1048576 bytes
dt (18126): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18128): Start Position 4294967296 (lba 8388608), Limit 1048576 bytes
dt (18128): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18115): Start Position 5368709120 (lba 10485760), Limit 1048576 bytes
dt (18115): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18133): Start Position 6442450944 (lba 12582912), Limit 1048576 bytes
dt (18133): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18134): Start Position 7516192768 (lba 14680064), Limit 1048576 bytes
dt (18134): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18132): Start Position 8589934592 (lba 16777216), Limit 1048576 bytes
dt (18132): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt (18131): Start Position 9663676416 (lba 18874368), Limit 1048576 bytes
dt (18131): Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
                .
                .
                .
Total Statistics (18138):
     Output device/file name: /dev/rrz1c (Device: BB01811C, type=disk)
     Type of I/O's performed: sequential (forward, read-after-write)
      Slice Range Parameters: position=11811160064 (lba 23068672), limit=1048576
      Current Slice Reported: 12/12
   Data pattern read/written: 0xffffffff (w/lbdata, lba 0, size 512 bytes)
     Total records processed: 8 @ 262144 bytes/record (256.000 Kbytes)
     Total bytes transferred: 2097152 (2048.000 Kbytes, 2.000 Mbytes)
      Average transfer rates: 1797559 bytes/sec, 1755.429 Kbytes/sec
     Number I/O's per second: 6.857
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m01.16s
           Total system time: 00m00.00s
             Total user time: 00m00.05s
               Starting time: Mon Jan 29 14:38:49 2001
                 Ending time: Mon Jan 29 14:38:51 2001
```

```
dt: Child process 18138, exiting with status 0
tru64%
```

The following test shows enabling variable requests sizes. Each request will be between the min and max values specified. Again, the debug is only enabled to show you the affects of this option.

```
tru64% dt of=/dev/rrz1c min=4k max=256k incr=variable enable=Debug,lbdata disable=pstats count=3
dt: Attempting to open output file '/dev/rrz1c', open flags = 01 (0x1)...
dt: Output file '/dev/rrz1c' successfully opened, fd = 3
dt: Allocated buffer at address 0x140036000 of 262148 bytes, using offset 0
dt: Record #1 (lba 0), Writing 4096 bytes from buffer 0x140036000...
dt: Record #2 (lba 8), Writing 235520 bytes from buffer 0x140036000...
dt: Record #3 (lba 468), Writing 225280 bytes from buffer 0x140036000...
dt: End of Write pass 0, records = 3, errors = 0, elapsed time = 00m00.05s
dt: Closing file '/dev/rrz1c', fd = 3...
dt: Attempting to reopen file '/dev/rrz1c', open flags = 00 (0)...
dt: File '/dev/rrz1c' successfully reopened, fd = 3
dt: Record #1 (lba 0), Reading 4096 bytes into buffer 0x140036000...
dt: Record #2 (lba 8), Reading 235520 bytes into buffer 0x140036000...
dt: Record #3 (lba 468), Reading 225280 bytes into buffer 0x140036000...
dt: End of Read pass 1, records = 3, errors = 0, elapsed time = 00m00.10s
dt: Closing file '/dev/rrz1c', fd = 3...

Total Statistics:
     Output device/file name: /dev/rrz1c (Device: BB01811C, type=disk)
     Type of I/O's performed: sequential (forward, rseed=0xf41e15)
   Data pattern read/written: 0x39c39c39 (w/lbdata, lba 0, size 512 bytes)
     Total records processed: 6 with min=4096, max=262144, incr=variable
     Total bytes transferred: 929792 (908.000 Kbytes, 0.887 Mbytes)
      Average transfer rates: 6198613 bytes/sec, 6053.333 Kbytes/sec
     Number I/O's per second: 40.000
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m00.15s
           Total system time: 00m00.00s
             Total user time: 00m00.06s
               Starting time: Mon Jan 29 14:30:08 2001
                 Ending time: Mon Jan 29 14:30:08 2001

tru64%
```

The next test shows the affect of enabling the reverse I/O direction on random access devices. The *capacity=* option artificially limits the size of the device media.

```
tru64% dt of=/dev/rrz1c bs=256k capacity=1m enable=Debug,lbdata,raw iodir=reverse
dt: Attempting to open output file '/dev/rrz1c', open flags = 02 (0x2)...
dt: Output file '/dev/rrz1c' successfully opened, fd = 3
dt: Random data limit set to 1048576 bytes (1.000 Mbytes), 2048 blocks.
```

```
dt: Allocated buffer at address 0x140036000 of 262148 bytes, using offset 0
dt: Allocated buffer at address 0x140078000 of 262148 bytes, using offset 0
dt: Seeked to block 2048 (0x800) at byte position 1048576.
dt: Seeked to block 1536 (0x600) at byte position 786432.
dt: Record #1 (lba 1536), Writing 262144 bytes from buffer 0x140078000...
dt: Seeked to block 1536 (0x600) at byte position 786432.
dt: Record #1 (lba 1536), Reading 262144 bytes into buffer 0x140036000...
dt: Seeked to block 1024 (0x400) at byte position 524288.
dt: Record #2 (lba 1024), Writing 262144 bytes from buffer 0x140078000...
dt: Seeked to block 1024 (0x400) at byte position 524288.
dt: Record #2 (lba 1024), Reading 262144 bytes into buffer 0x140036000...
dt: Seeked to block 512 (0x200) at byte position 262144.
dt: Record #3 (lba 512), Writing 262144 bytes from buffer 0x140078000...
dt: Seeked to block 512 (0x200) at byte position 262144.
dt: Record #3 (lba 512), Reading 262144 bytes into buffer 0x140036000...
dt: Seeked to block 0 (0) at byte position 0.
dt: Record #4 (lba 0), Writing 262144 bytes from buffer 0x140078000...
dt: Seeked to block 0 (0) at byte position 0.
dt: Record #4 (lba 0), Reading 262144 bytes into buffer 0x140036000...
dt: Beginning of media detected [file #1, record #4]
dt: Exiting with status code 254...
dt: Closing file '/dev/rrz1c', fd = 3...

Total Statistics:
     Output device/file name: /dev/rrz1c (Device: BB01811C, type=disk)
     Type of I/O's performed: sequential (reverse, read-after-write)
   Data pattern read/written: 0x39c39c39 (w/lbdata, lba 0, size 512 bytes)
     Total records processed: 8 @ 262144 bytes/record (256.000 Kbytes)
     Total bytes transferred: 2097152 (2048.000 Kbytes, 2.000 Mbytes)
      Average transfer rates: 9679163 bytes/sec, 9452.308 Kbytes/sec
     Number I/O's per second: 36.923
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 00m00.21s
           Total system time: 00m00.00s
             Total user time: 00m00.05s
               Starting time: Mon Jan 29 14:36:36 2001
                 Ending time: Mon Jan 29 14:36:37 2001

tru64%
```

The following example shows using the multiple volume options for us with removable media. Yea, this option is mainly for testing tapes, but testing with floppies was faster :-)

```
tru64% dt of=/dev/rfd0c bs=32k volumes=2 enable=lbdata vrecords=5 aios=4

Please insert volume #2 in drive /dev/rfd0c, press ENTER when ready to proceed:
    [ Continuing at record #46, bytes written so far 1474560... ]
```

```
Write Statistics:
     Total records processed: 50 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 1638400 (1600.000 Kbytes, 1.562 Mbytes)
      Average transfer rates: 8004 bytes/sec, 7.816 Kbytes/sec
     Number I/O's per second: 0.244
     Total volumes completed: 2/2
      Total passes completed: 0/1
       Total errors detected: 0/1
          Total elapsed time: 03m24.70s
           Total system time: 00m00.00s
             Total user time: 00m00.01s

Please insert volume #1 in drive /dev/rfd0c, press ENTER when ready to proceed:

Please insert volume #2 in drive /dev/rfd0c, press ENTER when ready to proceed:
    [ Continuing at record #46, bytes read so far 1474560... ]

Read Statistics:
     Total records processed: 50 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 1638400 (1600.000 Kbytes, 1.562 Mbytes)
      Average transfer rates: 13859 bytes/sec, 13.534 Kbytes/sec
     Number I/O's per second: 0.423
     Total volumes completed: 2/2
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 01m58.21s
           Total system time: 00m00.01s
             Total user time: 00m00.16s

Total Statistics:
     Output device/file name: /dev/rfd0c (Device: floppy, type=disk)
     Type of I/O's performed: sequential
   Data pattern read/written: 0x39c39c39 (w/lbdata, lba 0, size 512 bytes)
     Total records processed: 100 @ 32768 bytes/record (32.000 Kbytes)
     Total bytes transferred: 3276800 (3200.000 Kbytes, 3.125 Mbytes)
      Average transfer rates: 9373 bytes/sec, 9.153 Kbytes/sec
     Asynchronous I/O's used: 4
     Number I/O's per second: 0.286
     Total volumes completed: 2/2
      Total passes completed: 1/1
       Total errors detected: 0/1
          Total elapsed time: 05m49.61s
           Total system time: 00m00.01s
             Total user time: 00m00.18s
               Starting time: Mon Jan 15 13:56:41 2001
                 Ending time: Mon Jan 15 14:02:30 2001

tru64%
```

****

-END OF DOCUMENT-

Table of Contents