dt – Data Test Program
April 22nd, 2011
Version 17.39

Author: Robin T. Miller

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! ☺

Modification History

Date	Version	Description
April 22, 2011	17.38	Document multiple file/directory options, cache control options, corruption analysis, etc. Bumps major number to 17.xx ☺
July 27, 2009	16.20	Document retry/history, timing, and other minor changes.
January 4, 2007	15.32	Document alarm, *keepalive, noprogt, prefix, trigger, and slice options. Document noprog and timestamp flags. Update examples and more.
February 2, 2001	14.1	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.
November 10, 2000	13.22	Added more test features and options, including: read-after-write (raw), setting the random I/O seed (rseed), and multi-volume media testing.
July 24, 1999	12.0	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.
February 21, 1996	9.3	Documented iotype={random or sequential} option.
December 11, 1995	9.0	Logical block data feature, additional (higher) tty speeds, and other minor changes.
July 26, 1995	8.0	Modem testing, child process control, pattern string enhancements, and other minor changes.
September 11, 1993	7.0	Initial release of Users Manual.

Overview

dt is a generic data test program used to verify the proper operation of peripherals and I/O sub-systems, 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, 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 on Unix and you specify a raw disk device, dt will overwrite existing file systems, so please be careful! I HATE TO ADMIT, I'VE DONE THIS MYSELF!

Operating Systems Supported

dt is conditionalized to run on AIX, HP-UX, SUN, ULTRIX, OSF, QNX, SCO Unixware, Windows, and Linux operating systems. Porting is simple for OS's with POSIX API's.

Test Uses

Those people 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 sub-system.
- 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).
- Monitor slow or no I/O progress.
- Execute a trigger when failures occur.

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 is shown in Appendix A.

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.

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.

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.

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*=" and "*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.
or bs=random	Selects random size between 512 and 256k.

Log File "*log[tu]*=" Options

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).
- logt=filename will truncate the existing log file.
- logu=filename will create unique log files with multiple processes (w/pid).

Syntax:

log[tu]=filename The log file name to write.

Special format keywords are now expanded when part of the log file name, so unique names can be created for each test:

Log File Format Keywords:

%iodir = The I/O direction. %iotype = The I/O type.
%host = The host name. %pid = The process ID.
%user = The user name.

Example: log=dt_%host_%user_%iodir_%iotype-%pid.log

Please see the DiskTests.ksh script for examples of using this.

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.
- 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.

Keepalive Alarm Time "*alarm=*" Option

Keepalive Message "**keepalive=*" Options

These options control a user defined message that will be emitted during the test. The user defines how often to display the keepalive message, via the "*alarm=time*" option, and the format of the message(s), via the "**keepalive=string*" options. The normal "*keepalive=*" option defines the script emitted during the test, while "*pkeepalive=*" is the per pass message string, and "*tkeepalive=*" is the totals message string (overriding what *dt* normally displays).

Syntax:

```
alarm=time        The keepalive alarm time.
keepalive=string  The keepalive message string.
keepalivet=time   The keepalive message frequency.
pkeepalive=str    The pass keepalive msg string.
tkeepalive=str    The totals keepalive msg string.
```

Keepalive Message Format Control

The keepalive string is free format like a printf(), with the following format control strings:

```
Keepalive Format Control:
        %b = The bytes read or written.  %B = Total bytes read and written.
        %c = Record count for this pass. %C = Total records for this test.
        %d = The device name.            %D = The real device name.
        %e = The number of errors.       %E = The error limit.
        %f = The files read or written.  %F = Total files read and written.
        %h = The host name.              %H = The full host name.
        %k = The kilobytes this pass.    %K = Total kilobytes for this test.
        %l = Blocks read or written.     %L = Total blocks read and written.
        %m = The megabytes this pass.    %M = Total megabytes for this test.
        %p = The pass count.             %P = The pass limit.
        %r = Records read this pass.     %R = Total records read this test.
        %s = The seconds this pass.      %S = The total seconds this test.
        %t = The pass elapsed time.      %T = The total elapsed time.
        %i = The I/O mode (read/write)   %u = The user (login) name.
        %w = Records written this pass.  %W = Total records written this test.

Performance Keywords:
        %bps  = The bytes per second.    %lbps = Logical blocks per second.
        %kbps = Kilobytes per second.    %mbps = The megabytes per second.
        %iops = The I/O's per second.    %spio = The seconds per I/O.

  Lowercase means per pass stats, while uppercase means total stats.

  Default: %d Stats: mode %i, blocks %l, %m Mbytes, pass %p/%P, elapsed %t
                or if pass statistics summary is disabled:
           %d Stats: mode %i, blocks %L, %M Mbytes, pass %p/%P, elapsed %T
```

Here's an example used by Hazards' *diskdt* process:

```
keepalive="count = %C; e = %e; t = %S; IOpS = %IOPS; SpIO = %SPIO"
```

```
tkeepalive="STAT +RawMbytes %MBPS +RawReads %R +RawWrites %W";
```

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).
```

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, keep, or keeponerror}.

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.

Data Corruption Analysis Options

Several options have been added to improve data corruption analysis when using the IOT data pattern, the preferred pattern for maximum data validation.

Options Added:

```
boff=string      Set the buffer offsets to: dec or hex (Default: hex}
dfmt=string      Set the data format to: byte or word  (Default: byte}
maxbad=value     Set maximum bad blocks to display.    (default is 1)
enable=dumpall   Dump all blocks (good and bad).       (Default: disabled)
```

The file offset and block range of each corruption and good block is summarized, and the IOT data is analyzed to determine if it's from a different pass, indicating stale data. When the IOT pattern is selected, the data format is set to "dfmt=word", since IOT data is packed in 32-bit word values.

Dump Format "*dfmt*=" Option

This option controls the dump format used with corruption analysis performed on IOT data.

Syntax:
dfmt=string Set the data format to: byte or word (Default: word)

Buffer Offset "boff=" Option

This option controls the radix of the buffer offsets displayed with IOT corruption analysis.

Syntax:
boff=string Set the buffer offsets to: dec or hex (Default: hex)

Max Bad Blocks "maxbad=" Option

This option controls how many bad blocks to display during IOT corruption analysis. By default, all bad blocks will be displayed. The optional enable=dumpall will also dump good data blocks.

Syntax:
maxbad=value Set maximum bad blocks to display.

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.

Device Type "*dtype*=" Option

Input Device Type "*idtype*=" Option

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.

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 are 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.

File System Test Options

This is a summary of the directory/file options added:

```
            dir= (top level directory)
     1 to files= (for multiple files)
     1 to sdirs= (for multiple subdirectories)
     1 to depth= (subdirectory depth, nested subdirs)
   enable=fdebug (enable file system debugging only)
    dirp=string  The directory prefix for subdirs.
 maxdata=value  The maximum data limit (all files).
maxfiles=value  The maximum files for all directories.
bufmodes={buffered,unbuffered,cachereads,cachewrites}
                Set one or more buffering modes (Default: none)
enable/disable file system caching: (or use new format above)
    readcache        Read cache control.    (Default: enabled)
    writecache       Write cache control.   (Default: enabled)
    deleteperpass    Delete files per pass. (Default: disabled)


The files= option creates this many files in each directory.
Multiple procs= creates multiple top level directories w/PID.
The file system caching options enable or disable direct I/O.
```

File System Buffering Modes "bufmodes=" Option

This option allow you to control file system behavior. When multiple buffering modes are specified, dt round-robins the modes during each pass.

Syntax:

bufmodes={buffered,unbuffered,cachereads,cachewrites}
Set one or more buffering modes (Default: none)

This is an alternative to using the enable/disable{readcache,writecache} option.

Directory Path "dir=" Option

This option specifies the top level directory when creating multiple files/directories.

Special Notes:

- When specifying a directory path, the input/output file options do not require a path. The file name is appended to the directory path appropriately for Unix vs. WIndows.
- If the last part of the directory path does not exist, dt creates the directory (please use "mkdir -p" to create all paths). For example, dir="/var/tmp/dtdir" will create "dtdir" in /var/tmp, if it doesn't exist.
- When used in conjunction with multiple processes, the process ID is appended to the directory name, and the directory is created.
- When dispose=delete, after deleting files, the directory created is also deleted.
- The directory path, previously required via the dir=path option, is now optional, if you of= or if= options specify a directory path

- dt will detect the path, and automatically setup the directory path for you, making it easier to add multiple files/directories to existing scripts.

  Before: dir=/var/tmp of=dt.data (two parameters)
  Now: of=/var/tmp/dt.data gets broken into dir=/var/tmp of=dt.data

Syntax:
  dir=dirpath The directory path for files.

Directory Prefix "dirp=" Option

This option specifies a prefix to use with subdirectory names, overriding the default name of “dN”, where ‘N’ is the subdirectory name, e.g. “d0”.

Syntax:
  dirp=string The directory prefix for subdirs.

Number of Subdirectories "sdirs=" Option

This option specifies the number of subdirectories to create.

Special Notes:

- sdirs= is the subdirectory width, while depth= is the height which controls the number of nested subdirectories under each top level subdirectory.
- The default directory name prefix is "d", short so one can nest deeper without exceeding the path limit (Linix is 4096). But, dirprefix= allows you to override the direcotry prefix or set to "" for no prefix.
- Like the multiple files option, when exiting dt will delete all subdirectory files and directories, assuming dispose=delete.
- When sending dt signals, please use SIGINT or SIGTERM, so cleanup is possible. If you kill dt via SIGKILL (-9), then dt won't get a chance to do its' cleanup! Also remember, if dt created lots of files and directories, it may take awhile.
- File state is maintained, so file system full conditions read the same number of files and data as was written during the write pass

.

Syntax:
  sdirs=value The number of subdirectories.

Subdirectory Depth "depth=" Option

This option specifies the depth of subdirectories in each subdirectory created.

Syntax:
  depth=value The subdirectory depth.

File Limit "*files*=" Option

This option controls the number of disk or tape files to process.

Special Notes for Tapes:

- 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.

Special Notes for Disk Files:

- If dispose=delete, dt deletes all files when exiting.
- If dispose=keeponerror, signals are considered errors so files are kept.
- If a file system full condition is encountered, the write pass stops and the read pass will read as many files as were written. Subsequent passes will either overwrite previous file data, or you can include the truncate flag (oflags=trunc) to free space and loop on file system fills.
- When using a prefix string with the device name in its' format (%d), the prefix string is updated with each new file name to create uniqueness in each data block.
- When using the IOT data pattern, the file number is factored into the IOT pattern to generate unique data for each file, creating a different seed for each block.
- When dt's internal data patterns are used, the pass count and file number are used to cycle through dt's internal data patterns, again to create unique data per file.
- A postfix of "-NNNNNNNN" is added to each file name for uniqueness.
- Reference disable={readcache,writecache} to control file system caching.

Syntax:

files=value Set number of disk/tape files to process.

File Frequency Flush "ffreq=" Option

This option specifies how frequently to flush file system data in records. Normally the flush only happens are the end of the write pass, unless disable=fsync is specified.

This translates into a Unix fsync() or Windows FlushFileBuffers() to push data from the buffer cache to the underlying physical storage. This may be helpful during file system writes, to detect write failures that are otherwise unknown to dt due to their async nature. When write failures are missed, dt reports a data corruption during its' read/compare pass. This option may be used in lieu to Direct I/O, which does catch write failures, when buffered I/O is preferred

Syntax:
ffreq=value The frequency (in records) to flush buffers.

Maximum Data "maxdata=" Option

The maximum data created by all files can be specified via the maxdata= option. This is useful if you know the file system space, and wish to either divide this space between multiple test clients (NAS testing), or prevent dt from encountering a file system full condition (which is handled properly now).

Syntax:
maxdata=value The maximum data limit (all files).

Maximum Files "maxfiles=" Option

The maximim files created by in all directories can be specified via the maxfiles= option. This is useful if you don't wish to do the math to figure out how many files per directory and/or subdirectory are required to say create 1 million files.

Syntax:
maxfiles=value The maximum files for all directories.

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.

History "history=" Option

This option sets the number of I/O history entries to record. During failures, the history is dumped, which can be helpful when troubleshooting failures.

Syntax:

```
history=value    Set the number of history request entries.
```

History Data Size "hdsize=" Option

When I/O history is enabled, this option controls how many data bytes are saved for each I/O.

Syntax:

```
hdsize=value     Set the history data size (bytes to save).
                 Default hdsize=32 (set to 0 to disable copy)
```

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.
```

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}.
```

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}.
```

IOT Pass "*iotpass*=" Option

This option is used to specify the IOT pass number. When multiple passes occur, dt factors in the pass count to generate unique data during each pass. For example, the IOT seed is normally 0x01010101, and will be multiplied by the pass specified, useful for re-reading previously written IOT data patterns.

Syntax:
iotpass=value Set the IOT pattern for specified pass.

IOT Seed "*iotseed*=" Option

This option is used to specify the last IOT pattern seed dt used. When multiple passes occur, dt now factors in the pass count to generate unique data during each pass. For example, the IOT seed is normally 0x01010101, but this is now multiplied by the pass count for uniqueness.

Syntax:
iotseed=value Set the IOT pattern block seed value.

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.

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*=" and "*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.

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*=" and "*incr*=" control variable length record sizes.

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

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.

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.

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.
- When the "*step=value*" option is used, limit controls the maximum offset stepped to.

Syntax:

limit=value The number of bytes to transfer.

Capacity "capacity=" Option

This option allow one to set the disk capacity (for raw disks), to a particular capacity, rather than using the capacity returned by the host OS or using dt's automatic calculation logic, used during random and reverse I/O or with multiple slices options.

Syntax:

capacity=value Set the device capacity in bytes.
or capacity=max Set maximum capacity from disk driver.

Log File "log=" Option

This option redirects dt's output to a disk file, both standard error and standard output.

Syntax:

log[tu]=filename The log file name to write.
t=truncate, u=unique (w/pid)

Log File Format Control Strings

When specifying the log file, dt recognizes these control sequences:

Log File Format Keywords:

%iodir = The I/O direction. %iotype = The I/O type.
%host = The host name. %pid = The process ID.
%user = The user name.

Example: log=dt_%host_%user_%iodir_%iotype-%pid.log

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*. Finally, the last term used is DRD for *Distributed Request Dispatcher*.

Common Open Flags "*flags*=" Option

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,...}.
```

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.

No Progress Time "*noprogt*=" Option

This option allows you to specify a time (in seconds) to report when I/O is not making progress. This option is used in conjunction with the "*alarm*=" option to periodically check for an report when I/O is taking too long. This is especially useful during controller failover type testing.

Syntax:
noprogt=value Set the no progress time (in seconds).

Example:

```
dt … alarm=5s trigger="cmd:trigger" enable=noprog noprogt=120s
dt (16308): No progress made for 120 seconds!
dt (16308): Executing: trigger /var/tmp/dt.data-16308 noprog 512 131072 0 0 0
/var/tmp/dt.data-16308 noprog 512 131072 0 0 0
dt (16308): Trigger exited with status 2!
dt (16308): Sleeping forever...
  ...
```

In this example, an alarm() is set for every 5 seconds, and when the current I/O exceeds 120 seconds, a message is displayed and the trigger script is executed with "op = noprog". If the "*trigger*=" option were omitted, then only the warning message is displayed.

When the "*trigger*=*cmd*:..." option is utilized, the exit status controls the subsequent action to take: CONTINUE = 0, TERMINATE = 1, SLEEP = 2, or ABORT = 3

No Progress Time Trigger "*noprogtt*=" Option

This option allows you to specify a time (in seconds) when to initiate the no-progress time trigger script. Note: This option has no effect, unless the *noprogt*= option is enabled.

Syntax:
noprogtt=value Set the no progress time trigger (in seconds).

No Time "*notime*=" Option

This option allows you to disable timing of certain operations (system calls), when the no-progress options is enabled.

Special Notes:

- This option has no effect, unless the *noprogt*= option is enabled.
- `Valid optype's are: open close read write ioctl fsync msync aiowait`

Syntax:
notime=optype Disable timing of specified operation type.

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.
```

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 burnt myself this way. ☹]

Syntax:

passes=value The number of passes to perform.

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. Note: The 4 byte lba needs increased to 8 bytes for larger capacity disks!

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
```

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.

Prefix "*prefix*=" Option

This option allows the user to define a free format prefix string which is written at the beginning of each block. It is used to generate uniqueness useful when data corruption occur. Certain format control strings are interpreted as shown below.

Syntax:
prefix=string The data pattern prefix string.

The prefix format controls permitted are:

```
Prefix Format Control:
        %d = The device name.          %D = The real device name.
        %h = The host name.            %H = The full host name.
        %p = The process ID.           %P = The parent PID.
        %u = The user name.

  Example: prefix="%u@%h (pid %p)"
```

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 appended 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 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.

Set Queue Depth "*qdepth*=" Option

This option is currently only implemented on HP-UX. It allow you to set the queue depth of the device under test, overriding its' default. Note: The settings is sticky (retained).

Syntax:
qdepth=value Set the queue depth to specified value.

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.

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.

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.

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.

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.

Retry Delay "retry_delay=" Option

This option controls the number of seconds to delay between reads performed *after* a data corruption. (see enable=retryDC option)

Syntax:

```
retry_delay=value Delay before retrying operation. (Def: 5)
```

Slice "*slice*=" Option

This option is used with random access devices. This option is used in conjunction with the "*slices*=value" option, which divides the media into slices (see below), then "*slice=value*" defines the slice to do testing to. Since *dt* does the calculations, this simplifies simultaneous testing from multiple hosts to shared storage (usually a multi-initiator test requrement).

Syntax:

slice=value The specific disk slice to test.

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!

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.

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.

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

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 to be seeked past after each request.

Special Notes:
- The "*limit=value*" option can be used to set the maximum offset.

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

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.

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.

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=value*).
- 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).

Trigger Action "*trigger*=" Option

This option specifies a trigger action to take whenever an error occurs and/or when the no-progress time has been exceeded (see "*enable=noprog*"). It's main purpose is for triggering an anlyzer and/or stopping I/O by some means (panic, etc) when trouble-shooting.

Syntax:

trigger=type The trigger to execute during errors.

Trigger Types:
br = Execute a bus reset.
bdr = Execute a bus device reset.
seek = Issue a seek to the failing lba.
cmd:string = Execute command with these args:
string dname op dsize offset position lba errno

The first three options require Scu in your PATH.

When specifying the "cmd:" type, which invokes a program/script, the following arguments are passed on the command line:

Format: **cmd dname op dsize offset position lba errno noprogtime**

Where:
dname = The device/file name.
op = open/close/read/write/miscompare/noprog
dsize = The device block size.
offset = The current file offset.
position = The failing offset within block.
lba = The logical block address (relative for FS).
errno = The error number on syscall errors.
noprogtime = The no-progress time (in seconds).

Multiple Volumes "*volumes*=" Option

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.

Enable "*enable*=" and 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.

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:

- 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)

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)

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)

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 useful if your OS debugger does not support debugging child processes.

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

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)

Debug Output "*debug*" Flag

Verbose Debug Output "*Debug*" Flag

Other Debug Output "**debug*" Flags

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)
    edebug          End of file debug.     (Default: disabled)
    fdebug          File operations debug. (Default: disabled)
    pdebug          Process related debug. (Default: disabled)
    rdebug          Random debug output.   (Default: disabled)
    tdebug          Timer debug output.    (Default: disabled)
```

Delete Per Pass "*deleteperpass*" Flag

The deleteperpass option deletes test files between multiple passes. This is especially handy when testing file system full/quota exceeded conditions, to start clean on each pass. Otherwise, even with file truncation, subsequent passes may encounter early file system full conditions, esp. with random I/O, which may not be very interesting.

Flag:
deleteperpass Delete files per pass. (Default: disabled)

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 512 bytes of data (was 64, recently increased).

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)

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)

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)

History Dumping "*hdump*" Flag

This flag controls dumping the history entries at the end of a test. Normally dt only dumps the history during errors, but this option when enabled, dumps the history when exiting. This is useful if you are timing I/O's, or wish to see the LBA's I/O went to, etc.

Flag:
hdump History dump. (Default: disabled)

History Timing "*htiming*" Flag

This flag controls the timing of history entries. Please be aware, that enabling timing of each I/O will impact your overall test performance, as an extra system call is used to obtain system time.

Flag:
htiming History timing. (Default: disabled)

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)

Loop On Error "*looponerror*" Flag

This flag controls lopping on data corruption rereads. This can be helpful in capturing the failing read request on an analyzer.

Special Notes:
- Also see "retry_delay=value" and retryDC flag control.

Flag:
looponerror Loop on error. (Default: disabled)

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.
- Enabling this feature will degrade performance statistics (slightly).

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).

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)

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)

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)

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)

No I/O Progress "*noprog*" Flag

This flag controls whether *dt* will check for slow or no I/O progress during testing.

Special Notes:

- Enabling this flag will do nothing by itself. The "*alarm=*" option specifies the frequency of how often *dt* checks for no progress.
- The "*noprogt=secs*" option specified the no I/O progress time.
- If "*noprogt=*" is omitted, it defaults to the "*alarm=*" time value.
- The *noprog* flag is implicitly enabled by the "*noprogt=value*" option.

Flag:

noprog	No progress check. (Default: disabled)

Prefill "*prefill*" Flag

This flag controls the buffer prefill normally performed prior to reads. Normally, dt prefills the buffer with the inverted data pattern (1st four bytes). This, of course, ensures the data is overwritten with data read, but also imposes overhead not always desirable.

Special Notes:

- When IOT pattern is used, this flag is automatically enabled, since IOT blocks are unique.

Flag:
prefill Prefill read buffer. (Default: enabled)

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)

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)

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)

Retry Data Corruptions "*retryDC"* Flag

This flag controls whether a data corruption retry is performed. A second read is done to re-read the data, with direct I/O for file systems, and the data is compared against the previous read data, and the expected data. If the reread data matches the expected data, then dt assumes a "read failure" occurred, otherwise if the reread data matches the previous read, dt assumes a "write failure" (the data was written incorrectly).

Flag:

retryDC Retry data corruptions.(Default: enabled)

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)

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)

System Log "syslog" Flag

This flag controls logging startup/finish and errors being logged to the system logger. This can be helpful when correlating dt's errors with system (driver/file system) error messages.

Flag:
syslog Log errors to syslog. (Default: disabled)

Timestamp Blocks "*timestamp*" Flag

This flag controls whether blocks are timestamped when written. The timestamp is skipped during data comparisions, but *is* displayed if any remaining data is incorrect.

Special Notes:
- When IOT or lbdata patterns are used, the block number is overwritten by the timestamp.
- This flag is a stop-gap, until block tagging (w/more information) is implemented.

Flag:
timestamp Timestamp each block. (Default: disabled)

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)

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. *But beware, partial reads or writes of disk records if not at EOF is usually a problem!*

Flag:
verbose Verbose output. (Default: enabled)

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 di able 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*".

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.

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)

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)

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)

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)

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)

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)

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 long (value shown above is for 64-bit systems), or double for systems which don’t support “*long long*”.)

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'.

Future Enhancements?

Initially *dt* was written to be a generic test tool, designed to test any device, and although that was (mostly) accomplished, device specific tests needed to be and were developed, based on the device type detected or specified by the "*dtype*=" option if not determined automatically.

Some of the features requested include:

- Support for an initialization file (*.dtrc*) to setup frequent or common test parameters.
- Develop corruption analysis logic. What is this? Folks familiar with HP's Hazard know how valuable this is: data re-read logic, I/O history, metadata prowlers, and detailed analysis of expected and received data. A lot of work is involved here, especially with file system prowlers, which are responsible for converting file system data structures to physical underlying LBA's, to help identify bad data in analyzer traces.
- Improved file system testing. Although not developed as a file system exerciser, many folks use it this way. Multiple processes creating unique data files generates a data load, but many file system specific features, such as truncating files, file locking, creating lots of metadata (via subdirectories), and many more are not tested well. Major effort here!
- Supporting multiple devices in one *dt* invocation (perhaps a comma separated list). Although multiple processes or threads could accomplish this, it does add complexity requiring locking and switching to reentrant library API's, and the savings is shared code is minimal (I think) since most of the address space is data buffers.
- Multiple threads for I/O is likely to be implemented one day. The reason I haven't rushed this I/O method, is because POSIX AIO provides my need, and most modern day OS's now support POSIX AIO. Interestingly enough, the Linux AIO is implemented via POSIX threads! Threads are interesting to overcome OS's with a process limit, and threads (should) reduce system resources.
- Incorporate SCSI library to implement bus/target/lun reset triggers, etc.
- Interactive interface to keep the device open, like *scu* does, to allow more creative tests, especially for tapes and tape libraries (although most use *dt* for disk testing).
- Add output formats to allow statistics to be imported to tools such as MS Excel, etc.
- GUI front-end? Might be nice, but *not* necessary for test automation. Volunteers?
- Port to VMS? There's a need, so given the time, this will likely happen.
- Native Windows? Mostly there, thanks to the HP Hazard India team, but unfortuanatly I no longer have a Windows development environment, so I cannot supply updates. The code needs a few tweaks for file system testing, ported for raw I/O testing initially.
- iozone supplies many of the features above, so you may wish to consider this tool too. It's difficult, if not impossible, to supply sufficient features for everyones test needs! ☺

Final Comments

I'm happy to report that *dt* is getting wide spread use all over the world! Storage groups, terminal/lat groups, Q/A, developers, and other peripheral qualification groups are using *dt* as part of their testing. I guess maybe this will be my (computer) legacy? ☺

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*

(*OR HIRE ME AND ALLOW ME TO WORK REMOTELY FROM MESQUITE, NV?*)

***I'VE HEARD ENOUGH APPLAUSE! ☺**

Appendix A *dt* Help Text

The following help text is contained within the *dt* program. Please review the WhatsNew* files for the changes added for each version (the html/ directory has HTML versions of these files).

```
% dt help
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.
        dir=dirpath      The directory path for files.
        dirp=string      The directory prefix for subdirs.
        sdirs=value      The number of subdirectories.
        depth=value      The subdirectory depth.
        bs=value         The block size to read/write.
    or  bs=random        Random size between 512 and 256k.
        log[tu]=filename The log file name to write.
                         t=truncate, u=unique (w/pid)
        aios=value       Set number of AIO's to queue.
        alarm=time       The keepalive alarm time.
        keepalive=string The keepalive message string.
        keepalivet=time  The keepalive message frequency.
        pkeepalive=str   The pass keepalive message string.
        tkeepalive=str   The totals keepalive message string.
        align=offset     Set offset within page aligned buffer.
    or  align=rotate     Rotate data address through sizeof(ptr).
        capacity=value   Set the device capacity in bytes.
    or  capacity=max     Set maximum capacity from disk driver.
        bufmodes={buffered,unbuffered,cachereads,cachewrites}
                         Set one or more buffering modes (Default: none)
        boff=string      Set the buffer offsets to: dec or hex (Default: hex)
        dfmt=string      Set the data format to: byte or word (Default: word)
        dispose=mode     Set file dispose to: {delete, keep, or keeponerror}.
        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 disk/tape files to process.
        ffreq=value      The frequency (in records) to flush buffers.
        maxfiles=value   The maximum files for all directories.
        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}.
        iotpass=value    Set the IOT pattern for specified pass.
        iotseed=value    Set the IOT pattern block seed value.
        history=value    Set the number of history request entries.
        hdsize=value     Set the history data size (bytes to save).
        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.
    maxdata=value    The maximum data limit (all files).
    flags=flags      Set open flags:   {excl,sync,...}
    oflags=flags     Set output flags: {append,trunc,...}
    maxbad=value     Set maximum bad blocks to display.
    oncerr=action    Set child error action: {abort or continue}.
    nice=value       Apply the nice value to alter our priority.
    noprogt=value    Set the no progress time (in seconds).
    noprogtt=value   Set the no progress trigger time (secs).
    notime=optype    Disable timing of specified operation type.
    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.
    prefix=string    The data pattern prefix string.
    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.
    slice=value      The specific disk slice to test.
    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.
    stats=level      The stats level: {brief, full, or none}
    trigger=type     The trigger to execute during errors.
    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)
    deleteperpass    Delete files per pass. (Default: disabled)
    debug            Debug output.          (Default: disabled)
    Debug            Verbose debug output.  (Default: disabled)
    edebug           End of file debug.     (Default: disabled)
    fdebug           File operations debug. (Default: disabled)
    pdebug           Process related debug. (Default: disabled)
    rdebug           Random debug output.   (Default: disabled)
    tdebug           Timer debug output.    (Default: disabled)
    dump             Dump data buffer.      (Default: enabled)
    dumpall          Dump all blocks.       (Default: disabled)
    eof              EOF/EOM exit status.   (Default: disabled)
    fsalign          File system align.     (Default: disabled)
    funique          Unique output file.    (Default: disabled)
    fsync            Controls file sync'ing.(Default: runtime)
    header           Log file header.       (Default: enabled)
    hdump            History dump.          (Default: disabled)
    htiming          History timing.        (Default: disabled)
    lbdata           Logical block data.    (Default: disabled)
    logpid           Log process ID.        (Default: disabled)
```

```
    looponerror      Loop on error.          (Default: disabled)
    microdelay       Microsecond delays.     (Default: disabled)
    mmap             Memory mapped I/O.      (Default: disabled)
    multi            Multiple volumes.       (Default: disabled)
    noprog           No progress check.      (Default: disabled)
    prefill          Prefill read buffer.    (Default: enabled)
    pstats           Per pass statistics.    (Default: enabled)
    raw              Read after write.       (Default: disabled)
    readcache        Read cache control.     (Default: enabled)
    writecache       Write cache control.    (Default: enabled)
    retryDC          Retry data corruptions.(Default: enabled)
    sighup           Hangup signal control. (Default: enabled)
    stats            Display statistics.     (Default: enabled)
    syslog           Log errors to syslog.   (Default: disabled)
    timestamp        Timestamp each block.   (Default: disabled)
    trigargs         Trigger cmd arguments. (Default: enabled)
    unique           Unique pattern.         (Default: enabled)
    verbose          Verbose output.         (Default: enabled)
    verify           Verify data written.    (Default: enabled)

  Example: enable=debug disable=compare,pstats

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.
    direct (O_DIRECT)      Direct disk access. (don't cache data).
    fsync (O_FSYNC)        Sync both read/write data with disk file.
    rsync (O_RSYNC)        Synchronize read operations.
    sync (O_SYNC)          Sync updates for data/file attributes.
    large (O_LARGEFILE)    Enable large (64-bit) file system support.

Output Open Flags:
    append (O_APPEND)      Append data to end of existing file.
    dsync (O_DSYNC)        Sync data to disk during write operations.
    trunc (O_TRUNC)        Truncate an existing 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)
    retry_delay=value Delay before retrying operation. (Def: 5)

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 (4096 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.

Keepalive Format Control:
        %b = The bytes read or written.  %B = Total bytes read and written.
        %c = Record count for this pass. %C = Total records for this test.
        %d = The device name.            %D = The real device name.
        %e = The number of errors.       %E = The error limit.
        %f = The files read or written.  %F = Total files read and written.
        %h = The host name.              %H = The full host name.
        %k = The kilobytes this pass.    %K = Total kilobytes for this test.
        %l = Blocks read or written.     %L = Total blocks read and written.
        %m = The megabytes this pass.    %M = Total megabytes for this test.
        %p = The pass count.             %P = The pass limit.
        %r = Records read this pass.     %R = Total records read this test.
        %s = The seconds this pass.      %S = The total seconds this test.
        %t = The pass elapsed time.      %T = The total elapsed time.
        %i = The I/O mode (read/write)   %u = The user (login) name.
        %w = Records written this pass.  %W = Total records written this test.

Performance Keywords:
        %bps  = The bytes per second.    %lbps = Logical blocks per second.
        %kbps = Kilobytes per second.    %mbps = The megabytes per second.
        %iops = The I/O's per second.    %spio = The seconds per I/O.

  Lowercase means per pass stats, while uppercase means total stats.

  Default: %d Stats: mode %i, blocks %l, %m Mbytes, pass %p/%P, elapsed %t
                or if pass statistics summary is disabled:
           %d Stats: mode %i, blocks %L, %M Mbytes, pass %p/%P, elapsed %T

Log File Format Keywords:
        %iodir = The I/O direction.     %iotype = The I/O type.
        %host  = The host name.         %pid    = The process ID.
        %user  = The user name.

  Example: log=dt_%host_%user_%iodir_%iotype-%pid.log

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

Prefix Format Control:
        %d = The device name.           %D = The real device name.
        %h = The host name.             %H = The full host name.
        %p = The process ID.            %P = The parent PID.
        %u = The user name.

  Example: prefix="%u@%h (pid %p)"

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'.

    Trigger Types:
            br = Execute a bus reset.
            bdr = Execute a bus device reset.
            seek = Issue a seek to the failing lba.
            cmd:string = Execute command with these args:
              string dname op dsize offset position lba errno noprogt
              args following cmd:string get appended to above args.

            The first three options require Scu in your PATH.

    Defaults:
        errors=1, files=0, passes=1, records=0, bs=512, log=stderr
        pattern=0x39c39c39, dispose=delete, align=0 (page aligned)
        aios=8, dlimit=512, oncerr=continue, volumes=0, vrecords=1
        iodir=forward, iomode=test, iotype=sequential, stats=full
        iotseed=0x01010101, hdsize=32

    --> Date: April 18th, 2011, Version: 17.38, Author: Robin T. Miller <--
%
```

Appendix B Test Strategy

Depending on your needs, *dt* provides a wide range of options to help customize your tests. My preference is to use a variety of tests, and for that matter different test tools, since each tool has its' own strengths and I/O patterns. But in general, *dt* serves most of my needs. Here are a couple things to keep in mind while developing your test strategy:

- Are you testing storage, driver, firmware, switch, file systems, network, or all?
  - what are the buffer alignment restrictions (if any)?
  - what are the characteristics of the component (s)?
  - what are the debug capabilities (for trouble-shooting)?
  - what mechanisms are available to stop I/O on errors?
  - what is the best trigger mechanism? consider *scu* or *spt* if SCSI.
  - what tunables are available? queue depth, max transfer size, etc.
- What are your testing goals: stress testing, or reliability testing?
  - consider using a wide variety of variable request sizes.
  - consider using *runtime=* option to specify length of test times.
  - consider using *errors=* option to tolerate a number of errors.
- Are you concerned with buffer alignment and/or pattern sensitive data?
  - consider using *align=*, *pattern=*, and *pf=* options.
- Are you doing shared (multi-initiatior) style storage testing?
  - consider using *slices=* and *slice=* options to test sections of disks simultaneously from each host (an integral and necessary part of shared storage testing)
  - consider using *prefix=* to create unique string from each host.
- Before generating an I/O load, please consider the following:
  - what is the service queue limits of your storage (max I/O requests)?
  - what is the queue depth of your storage device, driver, and host adapter?
  - does your host OS disk driver handle "queue fulls" well?
  - how many hosts are accessing your shared storage?
  - how many processors, memory, and swap space is available?
  - depending out the above, you may need to limit your I/O loads.
- How much I/O load do you wish to generate?
  - consider *aios=*, *procs=*, and/or *slices=* options.
  - don't overdrive your host OS or storage (unless intended).
  - don't spawn so many processes that paging/swapping occurs.
- What tools are available for monitoring the I/O load?
  - consider using *iostat*, *vmstst*, *top*, etc to monitor I/O and processes.
  - consider monitoring per path I/O, e.g. AIX "*iostat –m*" w/MPIO.
  - consider monitoring/gathering statistics from your storage array.
  - consider using *scu* to gather SCSI Log Page statistics.
- Are you doing perturbation testing?
  - abort, bus/target/lun resets? consider using *scu* or *spt.*
  - do you need to do panic and reboot testing?

  - will you be doing storage controller failures?
  - consider failover characteristics of your storage.
  - consider using *alarm=* and *noprogt=* options to monitor I/O.
- Consider tools for troubleshooting problems:
  - does the OS supply an error logger? (AIX has *errpt*).
  - where do kernel error messages get written?
  - can you display kernel messages via *dmesg*?
  - does your host supply a method to panic the system?
  - consider using *trigger=* option to trigger analyzers or stop software traces.
- Are you doing performance testing?
  - use *aios=value* option with large value (say 64).
  - use larger block sizes: e.g. *bs=64k to 256k* or greater.
  - disable data comparisions via *disable=compar.*
  - keep buffers page aligned (i.e., don't use *align=* option).
  - do **not** use read-after-write (*enable=raw*) option.
  - sequntial I/O is always faster than random I/O (of course).
  - during file system testing, umount/re-mount to invalidate the buffer cache.
  - keep in mind, *dt* was not developed to be a perfomance tool (though useful).

Obviously, this is only some of what needs to be considered. Each storage device, host OS, drivers, etc. have different attributes. Each lab has their own requirements, and *dt* is usually wrapped by some test harness (Hazard, QSuite, NATE, etc). Sadly, none of these are open sourced nor productized for purchase, so test harnesses or scripts need to be developed.

Recommended Command Lines?

One of the most frequently asked questions, esp. with newbie's, is what are good *dt* command lines to test with? Well, as described above, this really depends on your test needs, but here are good tests to consider for disk testing:

```
      dt of=%s aios=%u min=%u max=%u incr=var pattern=iot prefix="%h on %d"
iotype={random|sequential}

      dt of=%s aios=%u min=%u max=%u incr=%u enable=lbdata prefix="%h on %d"
iotype={random|sequential}
```

If sequential I/O, vary this option: iodir={forward|reverse}
iodir=reverse starts at the end of the disk/file, and stops at the beginning.

For best performance:

```
dt of=%s aios=%u bs=%u (max block size supported by OS)
```

Disabling data comparisions improves performance.

Appendix C *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).

There's a Korn shell script in Scripts/DiskTests.ksh, for doing regression testing. If you are new to using *dt*, this script is a good starting point!

Simple Raw Test (to get started)

TEST DESCRIPTION: This test does read testing of a raw disk partition with data comparisons disabled using the POSIX Asynchronous I/O (8 by default).

```
% 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
%
```

Simple File System Test

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 test files created when 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
%
```

Memory Mapped File Test

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's. File data is mapped automaticically by the virtual memory system, rather than using read() or write() API's. Multiple passes will show much improved performance since the data is paged in/out or already in the buffer cache.

```
% 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
%
```

QIC Tape Test

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. Note: 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
%
```

Multiple Disk Files Test

TEST DESCRIPTION: This test writes then read/verifies 10 disk files, disabling read caching to ensure reads originate from the underlying storage rather than the file system cache. The files are kept, so a latter read-only pass can be performed using the same files.

```
robin-ptc% dt of=/var/tmp/dt_files/dt.data files=10 limit=10m dispose=keep
pattern=iot disable=pstats,readcache
dt: End of Write pass 0/1, 204800 blocks, 100.000 Mbytes, 20480 records,
errors 0/1, elapsed 00m02.83s
dt: End of Read pass 1/1, 204800 blocks, 100.000 Mbytes, 20480 records,
errors 0/1, elapsed 00m18.92s

Total Statistics:
     Output device/file name: /var/tmp/dt_files/dt.data-00000010 (device
type=regular)
     Type of I/O's performed: sequential (forward)
    Data pattern string used: 'IOT Pattern' (blocking is 512 bytes)
     Total records processed: 409600 @ 512 bytes/record (0.500 Kbytes)
     Total bytes transferred: 209715200 (204800.000 Kbytes, 200.000 Mbytes)
      Average transfer rates: 9642078 bytes/sec, 9416.092 Kbytes/sec
     Number I/O's per second: 18832.184
      Total passes completed: 1/1
       Total files processed: 20/20
       Total errors detected: 0/1
          Total elapsed time: 00m21.75s
           Total system time: 00m13.15s
             Total user time: 00m00.82s
               Starting time: Fri Apr 22 15:16:06 2011
                 Ending time: Fri Apr 22 15:16:28 2011

robin-ptc% ls -ls /var/tmp/dt_files
total 102560
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000001
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000002
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000003
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000004
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000005
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000006
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000007
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000008
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000009
10256 -rw-r--r-- 1 rtmiller gopher 10485760 Apr 22 15:16 dt.data-00000010
robin-ptc% dt if=/var/tmp/dt_files/dt.data files=10 limit=10m pattern=iot
disable=readcache

Total Statistics:
      Input device/file name: /var/tmp/dt_files/dt.data-00000010 (device
type=regular)
     Type of I/O's performed: sequential (forward)
```

```
    Data pattern string used: 'IOT Pattern' (blocking is 512 bytes)
     Total records processed: 204800 @ 512 bytes/record (0.500 Kbytes)
     Total bytes transferred: 104857600 (102400.000 Kbytes, 100.000 Mbytes)
      Average transfer rates: 5553898 bytes/sec, 5423.729 Kbytes/sec
     Number I/O's per second: 10847.458
      Total passes completed: 1/1
       Total files processed: 10/10
       Total errors detected: 0/1
          Total elapsed time: 00m18.88s
           Total system time: 00m12.61s
             Total user time: 00m00.70s
               Starting time: Fri Apr 22 15:17:23 2011
                 Ending time: Fri Apr 22 15:17:42 2011

robin-ptc%
```

Multiple Tape Files Test

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
%
```

Unix Pipe Testing

TEST DESCRIPTION: This test does writing/reading through a pipe. Notice the special character '-' which indicates write standard out/read standard in. While one might say why is this important, consider how *dt* can be used as a data generation tool for other programs! ☺

```
% 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
%
```

Unix FIFO Testing

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
%
```

Serial Line Testing

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 by default, although an option exists to add one.

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 overwrite the end of buffers (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...
%
```

Multiple Process Test

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
%
```

Tru64 Unix Disklabel Note

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
#
```

Data Corruption – Buffer Overrun Issue

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
%
```

Data Corruption – Tape Variable Record Issue

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...
%
```

Data Corruption – I/O Hang Issue

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\s-2
>  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]\s+2
(dbx) quit
%
```

Data Corruption – Tape Buffer Overrun Issue

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%
```

Another Use – Copy/Verify Data

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

%
```

Tru64 Unix Extended Error Information (EEI)

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

$
```

Multiple Volume Tape Test

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%
```

Read-After-Write Test

DESCRIPTION: This example shows the results of doing a read-after-write (raw) test to a floppy diskette. This option is valid with disks and tapes. This option avoids the multiple write and read pass *dt* normally performs. *Yes I know, **raw** is a bad option name! (sorry)*

```
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%
```

Slice And Dice Test

TEST DESCRIPTION: 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%
```

Variable I/O Requests Test

TEST DESCRIPTION: 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%
```

Reverse I/O Test

TEST DESCRIPTION: 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%
```

Multiple Volume Options Test

TEST DESCRIPTION: 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%
```

Data Corruption w/Timestamp Option

TEST DESCRIPTION: Forcing a data compare failure to show block timestamps. This option can be useful to determine when this data was written. With file systems, the data is oftentimes flushed from the buffer cache much later from when it was originally written. Unless Direct I/O (DIO) other open *SYNC flags are specified, *dt* only flushes data at the end of the write pass.

```
hyde% dt if=/var/tmp/dtc.data bs=64k records=100 dispose=keep pattern=iot
enable=timestamp iotseed=0x0d0d0d0d prefix="%d@%h"
dt: ERROR: Error number 1 occurred on Mon Apr 18 15:55:39 2011
dt: Elapsed time since beginning of pass: 00m00.21s
dt: Elapsed time since beginning of test: 00m00.21s
dt: Data compare error at byte 11 in record number 17
dt: Relative block number where the error occurred is 2048, position 1048587
(offset 11)
dt: Data expected = 0x63, data found = 0x78, byte count = 65536
dt: Mismatch of data pattern prefix: '/var/tmp/dtc.data@hyde'
dt: The correct data starts at address 0x545db (marked by asterisk '*')
dt: Dumping Prefix Buffer (base = 0x545d0, offset = 11, limit = 24 bytes):
  Address / Offset
0x000545d0/     0 | 2f 76 61 72 2f 74 6d 70 2f 64 74*63 2e 64 61 74 "/var/tmp/dtc.dat"
0x000545e0/    16 | 61 40 68 79 64 65 00 00                         "a@hyde  "

dt: The data block was written on Mon Apr 18 15:55:29 2011
dt: The incorrect data starts at address 0x6a00b (marked by asterisk '*')
dt: Dumping Data Buffer (base = 0x6a000, offset = 11, limit = 512 bytes):
  Address / Offset
0x0006a000/     0 | 2f 76 61 72 2f 74 6d 70 2f 64 74*78 2e 64 61 74 "/var/tmp/dtx.dat"
0x0006a010/    16 | 61 40 68 79 64 65 00 00 4d ac 97 31 00 00 00 00 "a@hyde  M  1    "
0x0006a020/    32 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 "                "
0x0006a030/    48 | 00 00 00 00 07 0f 07 07 08 10 08 08 09 11 09 09 "                "
0x0006a040/    64 | 0a 12 0a 0a 0b 13 0b 0b 0c 14 0c 0c 0d 15 0d 0d "                "
0x0006a050/    80 | 0e 16 0e 0e 0f 17 0f 0f 10 18 10 10 11 19 11 11 "                "
0x0006a060/    96 | 12 1a 12 12 13 1b 13 13 14 1c 14 14 15 1d 15 15 "                "
0x0006a070/   112 | 16 1e 16 16 17 1f 17 17 18 20 18 18 19 21 19 19 "             !  "
0x0006a080/   128 | 1a 22 1a 1a 1b 23 1b 1b 1c 24 1c 1c 1d 25 1d 1d " "   #   $   %  "
0x0006a090/   144 | 1e 26 1e 1e 1f 27 1f 1f 20 28 20 20 21 29 21 21 " &   '   (  !)!!"
0x0006a0a0/   160 | 22 2a 22 22 23 2b 23 23 24 2c 24 24 25 2d 25 25 ""*""#+##$,$$%-%%"
0x0006a0b0/   176 | 26 2e 26 26 27 2f 27 27 28 30 28 28 29 31 29 29 "&.&&'/''(0(()1))"
0x0006a0c0/   192 | 2a 32 2a 2a 2b 33 2b 2b 2c 34 2c 2c 2d 35 2d 2d "*2**+3++,4,,-5--"
0x0006a0d0/   208 | 2e 36 2e 2e 2f 37 2f 2f 30 38 30 30 31 39 31 31 ".6../7//08001911"
0x0006a0e0/   224 | 32 3a 32 32 33 3b 33 33 34 3c 34 34 35 3d 35 35 "2:223;334<445=55"
0x0006a0f0/   240 | 36 3e 36 36 37 3f 37 37 38 40 38 38 39 41 39 39 "6>667?778@889A99"
```

```
0x0006a100/   256 | 3a 42 3a 3a 3b 43 3b 3b 3c 44 3c 3c 3d 45 3d 3d ":B::;C;;<D<<=E=="
0x0006a110/   272 | 3e 46 3e 3e 3f 47 3f 3f 40 48 40 40 41 49 41 41 ">F>>?G??@H@@AIAA"
0x0006a120/   288 | 42 4a 42 42 43 4b 43 43 44 4c 44 44 45 4d 45 45 "BJBBCKCCDLDDEMEE"
0x0006a130/   304 | 46 4e 46 46 47 4f 47 47 48 50 48 48 49 51 49 49 "FNFFGOGGHPHHIQII"
0x0006a140/   320 | 4a 52 4a 4a 4b 53 4b 4b 4c 54 4c 4c 4d 55 4d 4d "JRJJKSKKLTLLMUMM"
0x0006a150/   336 | 4e 56 4e 4e 4f 57 4f 4f 50 58 50 50 51 59 51 51 "NVNNOWOOPXPPQYQQ"
0x0006a160/   352 | 52 5a 52 52 53 5b 53 53 54 5c 54 54 55 5d 55 55 "RZRRS[SST\TTU]UU"
0x0006a170/   368 | 56 5e 56 56 57 5f 57 57 58 60 58 58 59 61 59 59 "V^VVW_WWX`XXYaYY"
0x0006a180/   384 | 5a 62 5a 5a 5b 63 5b 5b 5c 64 5c 5c 5d 65 5d 5d "ZbZZ[c[[\d\\]e]]"
0x0006a190/   400 | 5e 66 5e 5e 5f 67 5f 5f 60 68 60 60 61 69 61 61 "^f^^_g__`h``aiaa"
0x0006a1a0/   416 | 62 6a 62 62 63 6b 63 63 64 6c 64 64 65 6d 65 65 "bjbbckccdlddemee"
0x0006a1b0/   432 | 66 6e 66 66 67 6f 67 67 68 70 68 68 69 71 69 69 "fnffgogghphhiqii"
0x0006a1c0/   448 | 6a 72 6a 6a 6b 73 6b 6b 6c 74 6c 6c 6d 75 6d 6d "jrjjkskkltllmumm"
0x0006a1d0/   464 | 6e 76 6e 6e 6f 77 6f 6f 70 78 70 70 71 79 71 71 "nvnnowoopxppqyqq"
0x0006a1e0/   480 | 72 7a 72 72 73 7b 73 73 74 7c 74 74 75 7d 75 75 "rzrrs{sst|ttu}uu"
0x0006a1f0/   496 | 76 7e 76 76 77 7f 77 77 78 80 78 78 79 81 79 79 "v~vvw wwx xxy yy"

Analyzing IOT Record Data: (Note: Block #'s are relative to start of record!)

                IOT block size: 512
        Total number of blocks: 128 (65536 bytes)
        Current IOT seed value: 0x0d0d0d0d
       Previous IOT seed value: 0x0c0c0c0c
     Start of corrupted blocks: 1
    Length of corrupted blocks: 1 (512 bytes)
  Corrupted blocks file offset: 1048576 (lba 2048)
         Start of good blocks: 2
        Length of good blocks: 127 (65024 bytes)
      Good blocks file offset: 1049088 (lba 2049)
    Number of corrupted blocks: 1
   Number of good blocks found: 127
   Number of zero blocks found: 0

                  File offset: 1048576
                Transfer count: 65536 (0x10000)
          Read buffer address: 6a000
         Pattern base address: 58000
                Prefix string: /var/tmp/dtc.data@hyde
                Prefix length: 24 bytes (0x18)
                         Note: Incorrect data is marked with asterisk '*'

                 Record block: 0
           Record block offset: 1048576 (lba 2048)
        Prefix string compare: incorrect
      Expected prefix string: /var/tmp/dtc.data@hyde
      Received prefix string: /var/tmp/dtx.data@hyde
        Block timestamp value: 0x3197ac4d
        Data block written on: Mon Apr 18 15:55:29 2011
        Expected block number: 2048 (0x00000800)
        Received block number: 0 (0x00000000)
      Seed detected at offset: 52 (0x34) (word 13, zero based)
     Data written during pass: 1
      Calculated block number: 2048 (0x00000800)
  Received data is from seed: 0x01010101 (stale data)

Byte Expected: address 58000              Received: address 6a000
0000 7261762f 706d742f 6374642f 7461642e * 7261762f 706d742f 7874642f 7461642e
     r a v /  p m t /  c t d /  t a d .    r a v /  p m t /  x t d /  t a d .
0010 79684061 00006564 00000800 0d0d150d * 79684061 00006564 3197ac4d 00000000
```

```
      y h @ a      e d                     y h @ a      e d  1     M
0020 1a1a221a 27272f27 34343c34 41414941 * 00000000 00000000 00000000 00000000
0030 4e4e564e 5b5b635b 68687068 75757d75 * 00000000 07070f07 08081008 09091109
0040 82828a82 8f8f978f 9c9ca49c a9a9b1a9 * 0a0a120a 0b0b130b 0c0c140c 0d0d150d
0050 b6b6beb6 c3c3cbc3 d0d0d8d0 dddde5dd * 0e0e160e 0f0f170f 10101810 11111911
0060 eaeaf2ea f7f7fff7 05050d04 12121a11 * 12121a12 13131b13 14141c14 15151d15
0070 1f1f271e 2c2c342b 39394138 46464e45 * 16161e16 17171f17 18182018 19192119
0080 53535b52 6060685f 6d6d756c 7a7a8279 * 1a1a221a 1b1b231b 1c1c241c 1d1d251d
0090 87878f86 94949c93 a1a1a9a0 aeaeb6ad * 1e1e261e 1f1f271f 20202820 21212921
00a0 bbbbc3ba c8c8d0c7 d5d5ddd4 e2e2eae1 * 22222a22 23232b23 24242c24 25252d25
00b0 efeff7ee fcfd04fb 0a0a1208 17171f15 * 26262e26 27272f27 28283028 29293129
00c0 24242c22 3131392f 3e3e463c 4b4b5349 * 2a2a322a 2b2b332b 2c2c342c 2d2d352d
00d0 58586056 65656d63 72727a70 7f7f877d * 2e2e362e 2f2f372f 30303830 31313931
00e0 8c8c948a 9999a197 a6a6aea4 b3b3bbb1 * 32323a32 33333b33 34343c34 35353d35
00f0 c0c0c8be cdcdd5cb dadae2d8 e7e7efe5 * 36363e36 37373f37 38384038 39394139
0100 f4f4fcf2 020209ff 0f0f170c 1c1c2419 * 3a3a423a 3b3b433b 3c3c443c 3d3d453d
0110 29293126 36363e33 43434b40 5050584d * 3e3e463e 3f3f473f 40404840 41414941
0120 5d5d655a 6a6a7267 77777f74 84848c81 * 42424a42 43434b43 44444c44 45454d45
0130 9191998e 9e9ea69b ababb3a8 b8b8c0b5 * 46464e46 47474f47 48485048 49495149
0140 c5c5cdc2 d2d2dacf dfdfe7dc ececf4e9 * 4a4a524a 4b4b534b 4c4c544c 4d4d554d
0150 f9fa01f6 07070f03 14141c10 2121291d * 4e4e564e 4f4f574f 50505850 51515951
0160 2e2e362a 3b3b4337 48485044 55555d51 * 52525a52 53535b53 54545c54 55555d55
0170 62626a5e 6f6f776b 7c7c8478 89899185 * 56565e56 57575f57 58586058 59596159
0180 96969e92 a3a3ab9f b0b0b8ac bdbdc5b9 * 5a5a625a 5b5b635b 5c5c645c 5d5d655d
0190 cacad2c6 d7d7dfd3 e4e4ece0 f1f1f9ed * 5e5e665e 5f5f675f 60606860 61616961
01a0 feff06fa 0c0c1407 19192114 26262e21 * 62626a62 63636b63 64646c64 65656d65
01b0 33333b2e 4040483b 4d4d5548 5a5a6255 * 66666e66 67676f67 68687068 69697169
01c0 67676f62 74747c6f 8181897c 8e8e9689 * 6a6a726a 6b6b736b 6c6c746c 6d6d756d
01d0 9b9ba396 a8a8b0a3 b5b5bdb0 c2c2cabd * 6e6e766e 6f6f776f 70707870 71717971
01e0 cfcfd7ca dcdce4d7 e9e9f1e4 f6f6fef1 * 72727a72 73737b73 74747c74 75757d75
01f0 04040bfe 1111190b 1e1e2618 2b2b3325 * 76767e76 77777f77 78788078 79798179

dt: Rereading and verifying record data using Direct I/O...
dt: Seeked to block 2048 (0x800) at offset 1048576
dt: Record #17 - Reading 65536 bytes (128 blocks) into buffer 0x7e000, lba's
2048 - 2175 (pos 1048576)
dt: Reread data matches previous data read, possible write failure!
dt: Command line to re-read the corrupted data:
dt: -> /u/rtmiller/Tools/dt.d-WIP/solaris-sparc/dt if=/var/tmp/dtc.data
bs=65536 count=1 position=1048576 prefix="/var/tmp/dtc.data@hyde" pattern=iot
iotseed=0x0d0d0d0d enable=timestamp flags=direct

Total Statistics:
      Input device/file name: /var/tmp/dtc.data (device type=regular)
     Type of I/O's performed: sequential (forward)
    Data pattern prefix used: /var/tmp/dtc.data@hyde
    Data pattern string used: 'IOT Pattern' (blocking is 512 bytes)
     Total records processed: 17 @ 65536 bytes/record (64.000 Kbytes)
     Total bytes transferred: 1114112 (1088.000 Kbytes, 1.062 Mbytes)
      Average transfer rates: 5305295 bytes/sec, 5180.952 Kbytes/sec
     Number I/O's per second: 80.952
      Total passes completed: 1/1
       Total errors detected: 1/1
          Total elapsed time: 00m00.21s
           Total system time: 00m00.01s
             Total user time: 00m00.11s
               Starting time: Mon Apr 18 15:55:39 2011
                 Ending time: Mon Apr 18 15:55:39 2011

hyde%
```

Large Capacity Disk Testing

TEST DESCRIPTION: This test shows a method to verify large TB sized disks. A number of options are used, here are the key ones:

- *slices=16* to divide the capacity into 16 sections, each with their own process
- *aios=16* to queue 16 requests at a time for improved performance.
- *step=4g* to seek after every record, thefore avoid writing every block.
- *iodir=reverse* to start at the end of sections then work forward.
- other options are left as an exercise to the reader! ☺

```
shaix11# ./dt                           \
         of=/dev/rhdisk12               \
         capacity=15t                   \
         slices=16                      \
         step=4g                        \
         disable=pstats                 \
         aios=16                        \
         oncerr=abort                   \
         min=b                          \
         max=256k                       \
         incr=var                       \
         align=rotate                   \
         pattern=iot                    \
         iodir=reverse                  \
         prefix='%d@%h'                 \
         alarm=5s                       \
         noprogt=10s                    \
         runtime=1h
```

Appendix D *Trigger Script*

The *trigger*= option allows and external program or script to get control when errors occurs and/or when the no-progress time *noprogt*= option is used. Note, the no-progress trigger time *noprogtt= option can also be used. This allows you to monitor the no-progess time, then trigger the script at a different (higher) time value.*

Below is an example trigger script (thanks AIX team!). Observe the exit status which controls *dt*'s action when this script exits.

```
#!/usr/bin/ksh
# /x/eng/locals/powerpc-ibm-aix5.1/test/bin                    #
# We get called with these parameters                          #
# dname op dsize offset position lba errno noprogtime          #
# Where:                                                       #
#       $1 dname = The device/file name.                       #
#       $2 op = open/close/read/write/miscompare/noprog        #
#       $3 dsize = The device block size.                      #
#       $4 offset = The current file offset.                   #
#       $5 position = The failing offset within block.         #
#       $6 lba = The logical block address (relative for FS).  #
#       $7 errno = The error number on syscall errors.         #
#       $8 noprogtime = The no progress time                   #
#                                                              #
# Capture and display these parameters                         #
my_name="dt_io_timeout.ksh"
dev_name=$1
operation=$2
dev_bk_sz=$3
off_set=$4
pos=$5
log_blk=$6
err_num=$7
no_prog_time=$8

echo "$my_name *#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
echo "The device name is $dev_name."
echo "The operation is $operation."
echo "The device block size is $dev_bk_sz."
echo "The offset, position and lba are $off_set, $pos, $log_blk."
echo "The errno is $err_num."
echo "The no progress time is $no_prog_time."
echo "$my_name *#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"

if [[ $operation = "noprog" ]]; then
# return code meanings                                         #
# CONTINUE = 0, TERMINATE = 1, SLEEP = 2, or ABORT = 3         #
        dt_io_timeout_rc=3
        echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
        echo "$my_name: I/O has exceeded the limit."
        echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
        # Now run something useful to display information - like host_info #
        echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
```

```
        echo "$my_name: Running host_info"
        echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
        /x/eng/locals/powerpc-ibm-aix5.1/test/bin/host_info
else
   dt_io_timeout_rc=0
fi
echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"
echo "The return code that $my_name is sending to dt is $dt_io_timeout_rc."
echo "*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#"

# Set the return code and exit #
exit $dt_io_timeout_rc
```