

Overview of Data Test Program (dt)

Robin T. Miller

This Software is Provided

By

Robin's Nest Software Inc.

752 Pinnacle Court

Mesquite, NV 89027

(702) 345-2236

WARNING!!!

This software **MUST BE CONSIDERED DESTRUCTIVE** to any data stored on the device under test. **NO** safety precautions are taken to ensure existing data is preserved during testing!!!

Please Note

DT is Open Source. You are free to copy and distribute it!
Please see copyright notice/disclaimer for details.

Introduction

- The data test program (dt), has become very popular over the last 22+ years! (showing my age 😊)
- This overview describes dt's history, device support, options, and various examples to help get you started with developing tests.

Short History of DT

- Originally developed while working at Compugraphic in the late 1980's initially for testing tape drives on SUN/OS.
- Written to replace Unix *dd/cmp* utilities.
- Command syntax is similar to Unix *dd*.
- Later disks, serial/parallel lines, and other device type support was added.
- Developed to verify device drivers and for device qualification (new hardware).
- Goal was to be generic yet flexible enough to customize tests.

Operating Systems Supported

- List of OS's supported:
 - ◆ AIX
 - ◆ HP-UX
 - ◆ Linux
 - ◆ Solaris
 - ◆ FreeBSD
 - ◆ Mac OS
 - ◆ SCO Unixware
 - ◆ Tru64 Unix
 - ◆ QNX
 - ◆ Windows (Cygwin and native)
 - ◆ most OS's supporting POSIX API's
- Features supported vary by OS (e.g., AIO, etc)

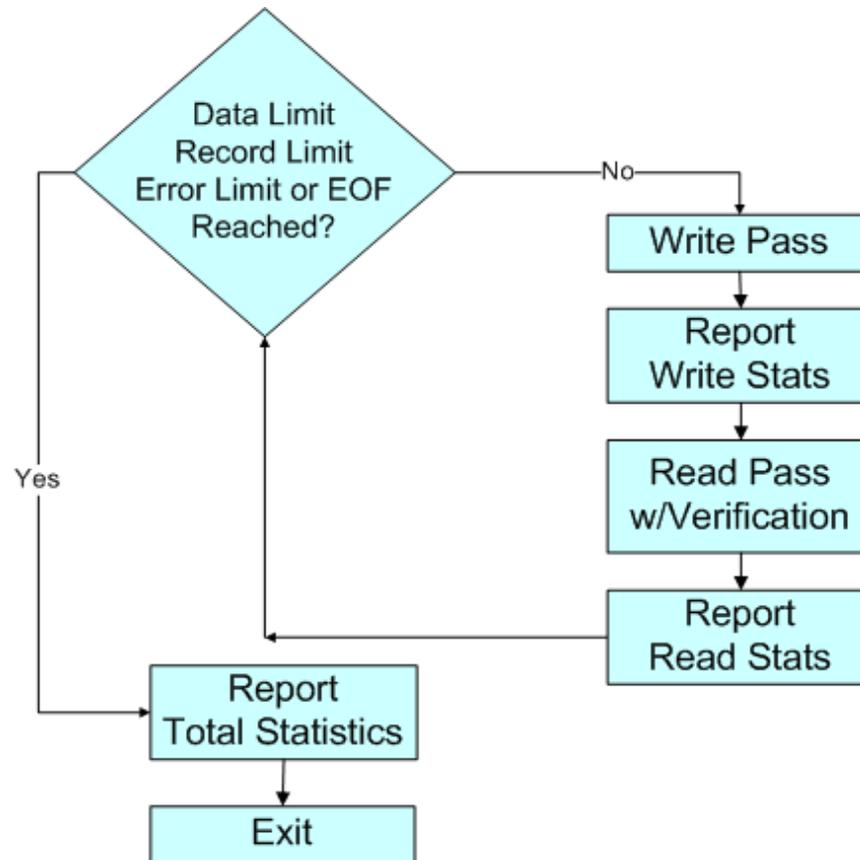
What Can Be Tested?

- Device drivers (disks, tapes, adapters, etc)
- File systems (UFS/VxFS/JFS/NFS, etc)
- Network interfaces (via rsh/ssh/NAS traffic)
- Virtual Memory via memory mapped files.
- Serial or parallel lines.
- Pipes (regular or named).
- Most any device capable of supporting standard open/read/write/close() API's.
- Exclude non-essential features by removing:
 - ◆ compilation flags: -DMMAP -DFIFO -DTTY

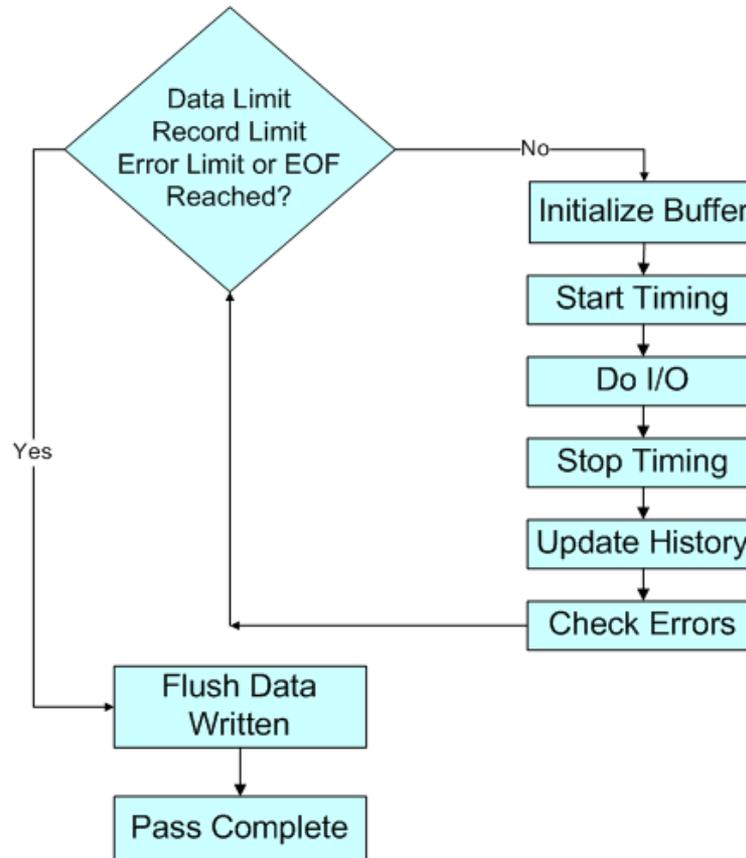
General Command Format

- To **write** then **read/verify** data:
dt of=filename bs=value limit=value ...
- The **read/verify** previously written data:
dt if=filename bs=value limit=value ...
- To **write** without doing read/verify pass:
dt of=filename bs=value disable=verify ...
- To **read** without doing data comparison:
dt if=filename bs=value disable=compare ...
- To specify the device type and device size:
dt of=filename dtype=type dsize=value ...
- Note: filename is a valid path for your OS.

General I/O Flow



General I/O Loop



Data Limit Controls

- For sequential, do I/O until data limit, record limit, runtime, or end of file / beginning of file/media is reached.
- For random, do I/O until data limit, record limit, or runtime is reached.
- Options:
 - ◆ limit=value – the data limit (in bytes)
 - ◆ records=value – the record limit
 - ◆ capacity=value – user set capacity

Data Transfer Sizes

- `bs=value` – sets a fixed block size
- `min=value` – sets the minimum size
- `max=value` – sets the maximum size
- `incr=value` – sets the increment size
 - ◆ `incr=var` enables variable increments between the min/max values
- `dsize=value` – sets the device size
 - ◆ default is 512 bytes for disks
 - ◆ obtained via IOCTL on some OS's

Controlling Execution Time

- Normally each pass is controlled by the capacity or one of these options:
 - ◆ `limit=value` – Specifies the data limit.
 - ◆ `records=value` – Specifies record count.
 - ◆ for sequential I/O, stop at EOF/EOM.
 - ◆ for random I/O, loop until limit reached.
- `runtime=time` – Runtime in seconds.
 - ◆ Careful: Short runtimes may prevent read/verify pass depending on limits.
- `passes=value` – Specifies pass limit.

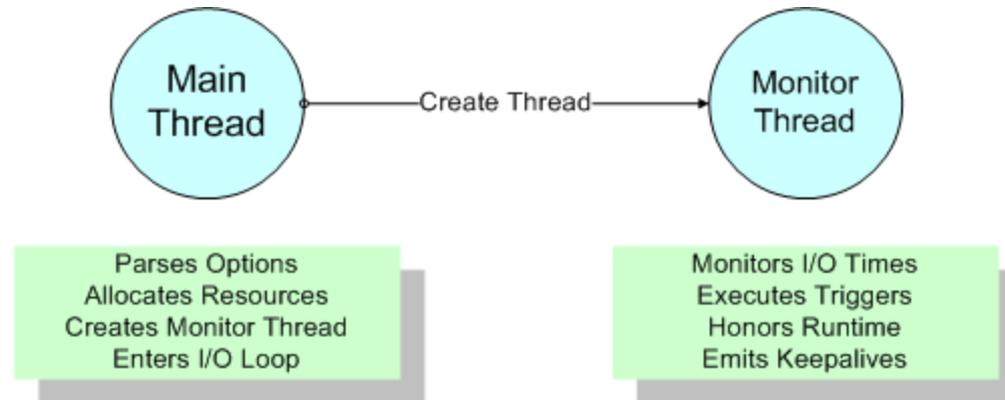
Program Features

- Data buffers are page aligned by default
 - ◆ align=value to misalign (e.g. align=rotate)
- Pad bytes are allocated for data buffers
 - ◆ Initialized with inverted pattern (1st 4 bytes)
 - ◆ On reads, verified after normal data
- Capacity is set via seek/reads or IOCTL
 - ◆ capacity=value – sets user capacity
- Data verification is enabled by default

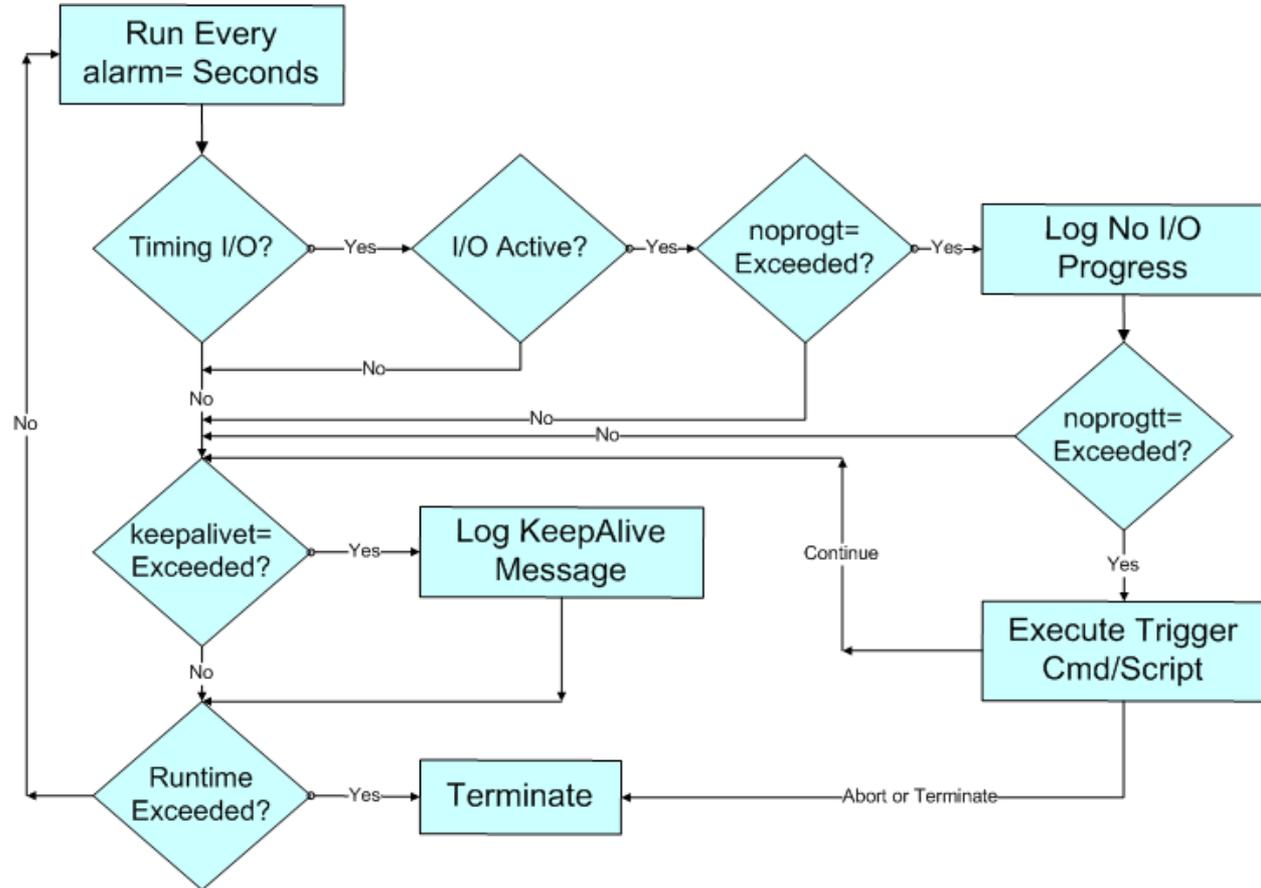
Unique Features

- Report no or slow I/O progress
 - ◆ can disable timing individual system calls
- Retries data corruptions
 - ◆ for file systems via Direct I/O (DIO)
 - ◆ may also loop on corruption with interval
- History for recording I/O's
 - ◆ data bytes saved
 - ◆ optionally with timing
- Log test info/errors to system logger
- Report periodic statistics via keepalive
- Trigger mechanism for troubleshooting

Process Threads



Monitor Thread



Numeric Input Formats

- w = words (4 bytes)
- b = blocks (512 bytes)
- m = megabytes
- g = gigabytes
- t = terabytes
- inf or INF = Infinite
- q = quadwords (8 bytes)
- k = kilobytes (1024 bytes)
- p = page size (varies)
- Simple arithmetic allowed.
- Bitwise operations allowed.
- use 'expr' for complex math.

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 '0' for octal conversions. Please Note: Evaluation is from right to left without precedence, and parenthesis are not permitted.

Time Input Format

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'.

Example:

```
# dt ... alarm=10s noprogt=1m runtime=24h30m15s
```

Pattern String Input

- pattern=string format controls supported:

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
```

Example:

```
# dt of=- pattern="Robin's test message\n\a" count=1 stats=none | head -2
Robin's test message
Robin's test message
#
```

Generic DT Features

- Assortment of data patterns (pattern= or pf= options).
- Control of I/O request sizes (min=, max=, incr=, bs= opts).
- Checks for buffer overwrites (pad bytes w/inverted pattern).
- Allows misaligned data buffers (align= option vs. page aligned).
- Allows block number encoding (enable=lbdata or pattern=iot).
 - ◆ LBA's are 32-bits
 - ◆ lbdata is big-endian LBA's
 - ◆ iot encodes little-endian LBA's
- Reports performance statistics (total stats after test complete).
- Variable I/O request sizes (incr=variable option).
- Various flags to control program behavior (debug, verify, etc).
- Intuitive numeric and runtime string input formats.
- Flags to control test operation (flags=direct,excl,sync etc).
- No I/O progress options to detect and report slow or no I/O.

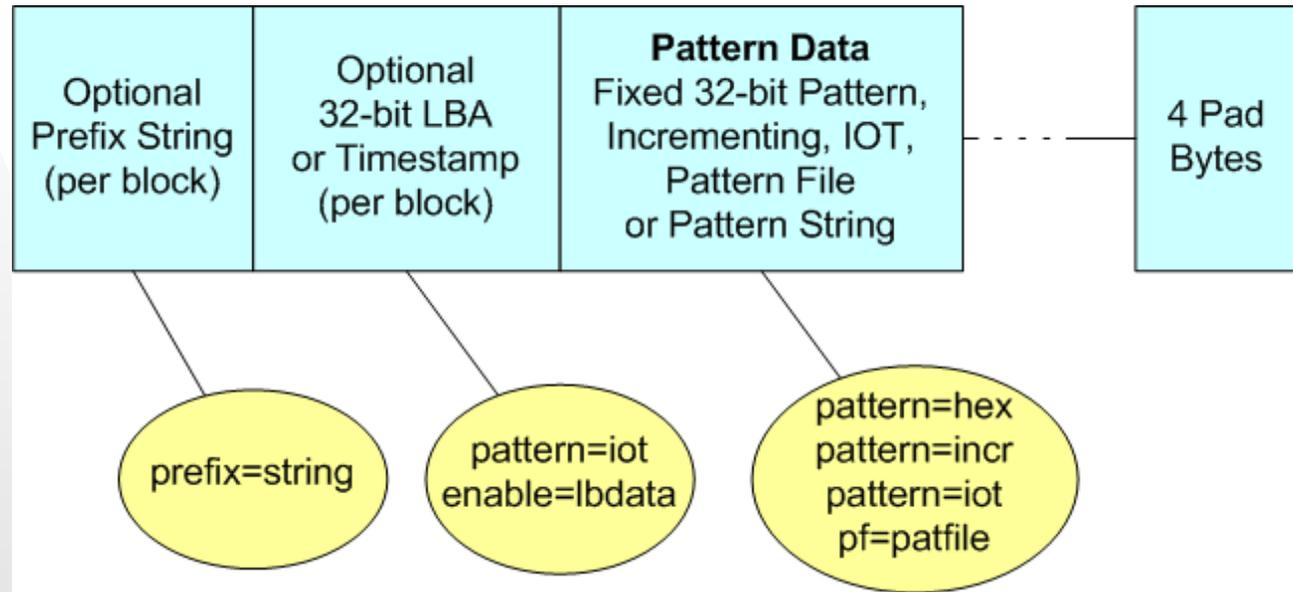
Generic DT Options

- Limit I/O by data limit (limit= option).
- Limit I/O by record limit (count= or records= options).
- Limit I/O by volume limit (volumes= and vrecords= options).
- Limit I/O by runtime limit (runtime=timeString option).
- Control of error limit (errors= option, default is 1).
- Control of pass limit (passes= option, default is 1).
- Control of multiple processes (procs= option).
- Control of process behavior (oncerr=abort or **continue**).
- Control of device types, size, etc. (dtype= and dsize= options).
- POSIX Asynchronous I/O (enable=aio or aios=value options).

Data Patterns

- 32 bit hex data pattern (pattern=value).
 - ◆ 13 internal data patterns cycled through for each write/read pass.
- IOT test pattern (pattern=iot).
- Incrementing data pattern:
 - ◆ 0, 1, 2, ... 255 (pattern=incr)
- Pattern string (pattern="string")
- Pattern file (pf=filename)
 - ◆ whole file read into memory

Data Pattern Layout



Internal Data Patterns

- Data patterns cycled through for each pass:
 - ◆ 0x39c39c39U
 - ◆ 0x00ff00ffU
 - ◆ 0x0f0f0f0fU
 - ◆ 0xc6dec6deU
 - ◆ 0x6db6db6dU
 - ◆ 0x55555555U
 - ◆ 0xaaaaaaaaU Complement of previous data pattern.
 - ◆ 0x33333333U Continuous worst case pattern (media defects)
 - ◆ 0x26673333U Frequency burst worst case pattern #1.
 - ◆ 0x66673326U Frequency burst worst case pattern #2.
 - ◆ 0x71c7c71cU Dibit worst case data pattern.
 - ◆ 0x00000000U
 - ◆ 0xffffffffU
- Multiple slices also cycle through these data patterns.

Data Pattern Files

- There are several predefined data pattern files included in the source kit, containing the following data patterns:
 - ◆ pattern_0 - psuedo-random pattern.
 - ◆ pattern_1 - psuedo-random pattern alternating with 0's.
 - ◆ pattern_2 - all 0s.
 - ◆ pattern_3 - all 1s (FFH).
 - ◆ pattern_4 - 16 bit shifting 1 in a field of 0s.
 - ◆ pattern_5 - 16 bit shifting 0 in a field of 1s.
 - ◆ pattern_6 - alternating 01 pattern (AAH).
 - ◆ pattern_7 - byte incrementing (OOH - FFH).
 - ◆ pattern_8 - encrypted data pattern.
 - ◆ pattern_9 - psuedo-random pattern1 alternating with 0's.
 - ◆ pattern_all – concatenation of all of the above patterns.
 - ◆ pattern_dedup - each pattern file written (bs=4k records=5).
- These files are specified with the pattern file (**pf=file**) option.

IOT Data Pattern

- The Logical Block Address (LBA) is encoded in the first 4 bytes of each data block (little-endian byte ordering).
- The constant value 0x01010101 is added to the LBA as a seed for subsequent words to generate uniqueness throughout the entire data block.
- Seed multiplied by pass count so unique for each pass.

Example:

```
# dt of=- count=1 pattern=iot stats=none | od -X
0000000 00000000 01010101 02020202 03030303
0000020 04040404 05050505 06060606 07070707
...
0000740 78787878 79797979 7a7a7a7a 7b7b7b7b
0000760 7c7c7c7c 7d7d7d7d 7e7e7e7e 7f7f7f7f
0001000
#
```

Open Flags

- **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 I/O (bypasses the buffer cache).
 - ◆ `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.
 - ◆ `defer (O_DEFER)` Defer updates to file during writes.
 - ◆ `dsync (O_DSYNC)` Sync data to disk during write operations.
 - ◆ `trunc (O_TRUNC)` Truncate an existing file before writing.

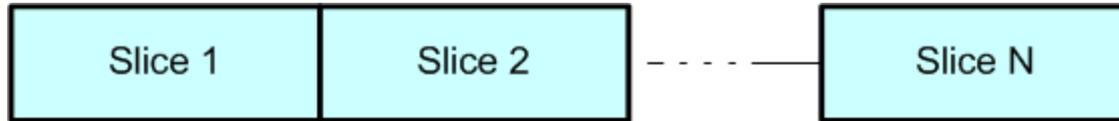
Example:

```
# dt of=dt.data flags=direct oflags=trunc passes=3 ...
```

Disk Specific Features

- Random & Sequential I/O (iotype=**sequential** or random)
- Supports raw & block devices (gleaned from stat() or IOCTL)
- Multiple procs creates multiple files (unique files using child PID)
- Supports copy/verify of disks (iomode=copy, **test**, or verify)
 - ◆ copy/verify is for image mode copy operations
- Supports reverse direction (iodir=**forward** or reverse)
 - ◆ reverse I/O is a variant of random I/O
- Supports multiple disk slices (slices= options)
 - ◆ shared LUN testing via slices=hosts slice=host#
- Supports skipping blocks (step=value option)
 - ◆ creates holes (empty space) in FS file. (aka sparse)
- Control FS output file disposition (dispose=**delete** or keep or keeponerror). Note: Has no affect on existing files! (always keeps)

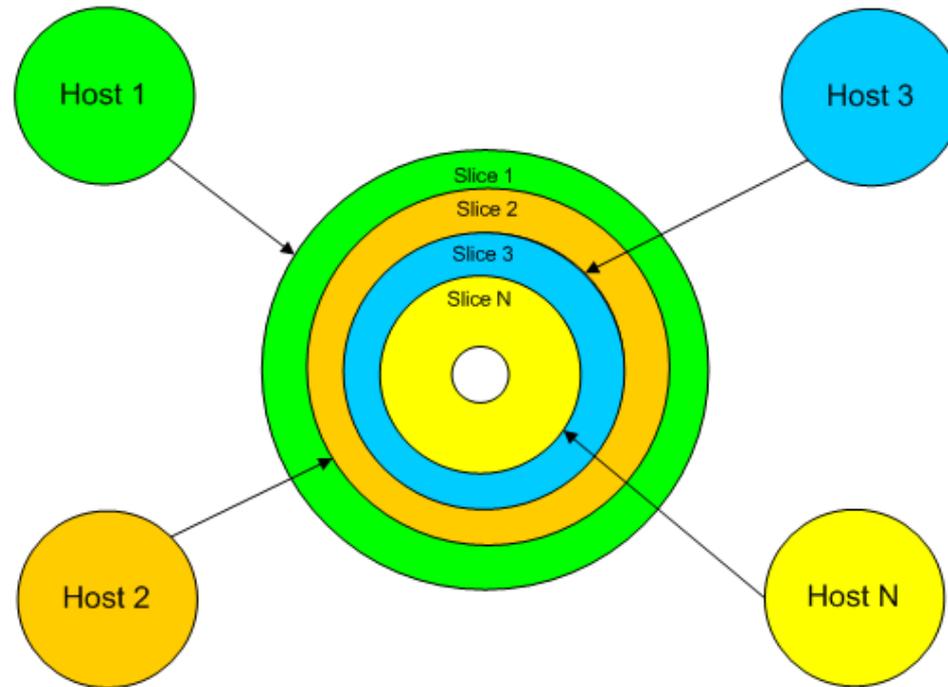
Slice and Dice



Example:

```
shaix11# dt of=/dev/rhdisk7 slices=3 slice=2 enable=debug disable=stats
dt (491562): IOCINFO Capacity: 2097152 blocks, device size 512 bytes.
dt (491562): Data limit set to 1073741824 bytes (1024.000 Mbytes), 2097152
blocks.
dt (491562): Slice 2 Information:
           Start: 357913600 offset (lba 699050)
           End: 715827200 offset (lba 1398100)
           Length: 357913600 bytes (699050 blocks)
           Limit: 357913600 bytes (699050 blocks)
dt: Attempting to open output file '/dev/rhdisk7', open flags = 01 (0x1)...
dt: Output file '/dev/rhdisk7' successfully opened, fd = 3
dt: Data limit set to 1073741824 bytes (1024.000 Mbytes), 2097152 blocks.
dt: Allocated buffer at address 0x20007000 of 516 bytes, using offset 0
dt: Using data pattern 0x39c39c39 for pass 1
dt: Closing file '/dev/rhdisk7', fd = 3...
dt: Attempting to reopen file '/dev/rhdisk7', open flags = 0 (0)...
dt: File '/dev/rhdisk7' successfully reopened, fd = 3
dt: Closing file '/dev/rhdisk7', fd = 3...
shaix11#
```

Shared LUN Testing



dt slices=N slice=1-N, so each host does I/O to its' own slice.

File System Features

- dir=dirpath – top level directory
- files=value – files per directory
- sdirs=value – subdirs per directory
- depth=value – subdirectory depth
- enable=fdebug – file system debugging
- procs=value – each process gets own tree
 - ◆ PID appended to directory path
- More details [here](#)

File System Features

- Ideas taken from IBM Blast tool:
 - ◆ maxdata=value – max data for all files
 - ◆ maxfiles=value – max files for all dirs
- More details [here](#)

File System Caching

- Beware of host OS buffer cache
- How to overcome?
 - ◆ reducing buffer cache size
 - ◆ truncate files (reallocates blocks)
 - ◆ Direct I/O (DIO) (bypass cache)
 - ◆ remount FS (invalidates cache)
 - ◆ alternate between:
 - ★ normal I/O, DIO, MMAP I/O

Troubleshooting Aids

- enable=flag options:
 - ◆ debug – 1st level debug (open, etc)
 - ◆ Debug – full debug output (verbose)
 - ◆ edebug – end of file/media debug
 - ◆ fdebug – file system debug
 - ◆ rdebug – random I/O debug
 - ◆ tdebug – timer debug output
 - ◆ syslog – write messages to syslog
 - ◆ timestamps – timestamp blocks
- dispose=keep – don't delete FS files
- history=value – I/O history mechanism
- trigger=type – specifies trigger type

History Mechanism

- `history=value` (disabled by default)
 - ◆ Sets the number of history request entries
- `hdsize=value` (default is 32 bytes)
 - ◆ Set the history data size (data bytes to save)
- `enable=hdump` (default is disabled)
 - ◆ Dumps history entries when exiting
- `enable=htiming` (default is disabled)
 - ◆ Enables timing history entries

Example History Output

- Options: history=5 enable=hdump,htiming

dt: Dumping History Data (5 entries):

```
1221852698.397315 (0.004768) Record #759 - Read 105472 bytes (206 blocks) lba's 204594 - 204799 (pos 104752128)
Offset
000000 00031f32 01042033 02052134 03062235
000016 04072336 05082437 06092538 070a2639
```

```
1221852698.392547 (0.002549) Record #758 - Read 240640 bytes (470 blocks) lba's 204124 - 204593 (pos 104511488)
Offset
000000 00031d5c 01041e5d 02051f5e 0306205f
000016 04072160 05082261 06092362 070a2463
```

```
1221852698.389998 (0.004830) Record #757 - Read 119808 bytes (234 blocks) lba's 203890 - 204123 (pos 104391680)
Offset
000000 00031c72 01041d73 02051e74 03061f75
000016 04072076 05082177 06092278 070a2379
```

```
1221852698.385168 (0.004725) Record #756 - Read 242688 bytes (474 blocks) lba's 203416 - 203889 (pos 104148992)
Offset
000000 00031a98 01041b99 02051c9a 03061d9b
000016 04071e9c 05081f9d 0609209e 070a219f
```

```
1221852698.380443 Record #755 - Read 231424 bytes (452 blocks) lba's 202964 - 203415 (pos 103917568)
Offset
000000 000318d4 010419d5 02051ad6 03061bd7
000016 04071cd8 05081dd9 06091eda 070a1fdb
```

Error Triggers

- Option: trigger=type
- Executed on failures or no-progress
- Trigger Types:
 - ◆ br = Execute a bus reset.
 - ◆ bdr = Execute a bus device reset.
 - ◆ seek = Issue a seek to the failing lba.
 - ◆ cmd:string = Execute string w/these args:
string dname op dsize offset position lba errno noprogt

The first three options require Scu in your PATH.
(will replace this with SCSI library in the future)

Trigger Script Arguments

- string dname op dsize offset position lba errno noprogt
 - ◆ string = executable program or script
 - ◆ dname = The device or file name
 - ◆ op = The operation in error
 - ◆ dsize = The device size
 - ◆ offset = The 64-bit file offset
 - ◆ position = Position in block of data corruption (DC)
 - ◆ lba = The logical block address (real or relative LBA)
 - ◆ errno = The error number
 - ◆ noprogt = The no-progress time value (in seconds)

Trigger Script Exit Value

- Trigger script exit value action (only pertains to no-progress I/O feature):
 - ◆ 0 = continue
 - ◆ 1 = terminate
 - ◆ 2 = sleep
 - ◆ 3 = abort

Simple Example:

```
% cat trigger
echo $*
exit 2
% dt of=/var/tmp/dt.data alarm=3s trigger="cmd:trigger" disable=stats
  flags=direct bs=32k noprogt=1s limit=1m procs=100
dt (16308): No progress made for 2 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...
...
```

- Example [no-progress trigger script](#).

Trigger Op Strings

- When executing a trigger script, the **op** argument is set to one of the following:
 - ◆ open – device open failed
 - ◆ close – close operation failed
 - ◆ read – read operation failed
 - ◆ write – write operation failed
 - ◆ noprog – no progress time exceeded
 - ◆ miscompare – data compare error
 - ◆ <various tape operations>, i.e. rewind, etc

Log File Strings

- Used with log="format-string":
 - ◆ %iodir = The I/O direction
 - ◆ %iotype = The I/O type
 - ◆ %host = The host name
 - ◆ %pid = The process ID
 - ◆ %user = The user (login) name
- Example: log="dt-%iodir-%iotype.log"
- Recently added, may be augmented.

Prefix Strings

- Prefix strings copied to start of each block.
- Used to create uniqueness between hosts and/or devices/files.
- Prefix Format Control Strings:

`%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)"`

KeepAlive Messages

- Originally added for HP's Hazard to avoid PTO's.
- Emits user defined message during long test runs.
- Allows you to roll your own pass/total messages.
- alarm=time controls frequency of keepalive msgs.
- keepalive=string - defines the runtime message.
- pkeepalive=string - defines per pass message.
- tkeepalive=string - defines the totals message.
- keepalivet=time - keepalive message frequency.
- Used when brief statistics are enabled (stats=brief):

Per Pass Default:

%d Stats: mode %i, blocks %l, %m Mbytes, pass %p/%P, elapsed %t

Totals Default:

%d Totals: %L blocks, %M Mbytes, errors %e/%E, passes %p/%P, elapsed %T

Keepalive Format Control

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

Performance Keywords:

<code>%bps</code> = The bytes per second.	<code>%lbps</code> = Logical blocks per second.
<code>%kbps</code> = Kilobytes per second.	<code>%mbps</code> = The megabytes per second.
<code>%iops</code> = The I/O's per second.	<code>%spio</code> = The seconds per I/O.

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

Keepalive Example

- Keep alive strings used by Hazards' dt disk wrapper:

```
% dt of=dt.data bs=32k limit=1g disable=pstats,verbose stats=brief \
    keepalive="count = %C; e = %e; t = %S; IOpS = %IOPS; SpIO = %SPIO" \
    tkeepalive="STAT +RawMbytes %MBPS +RawReads %R +RawWrites %W" \
    alarm=15s runtime=1m3s
dt: count = 32768; e = 0; t = 15; IOpS = 2184.533; SpIO = 0.0005
dt: count = 45077; e = 0; t = 30; IOpS = 1502.567; SpIO = 0.0007
dt: count = 80049; e = 0; t = 45; IOpS = 1778.867; SpIO = 0.0006
dt: count = 98304; e = 0; t = 60; IOpS = 1638.400; SpIO = 0.0006
dt: STAT +RawMbytes 46.574 +RawReads 46242 +RawWrites 65536
%
```

- External tools/scripts can monitor statistics during runs.
- Note: alarm option also used by I/O monitoring feature.
 - ◆ `keepalivet=value` – Controls keepalive time (in seconds).

Special Notes

- Redirect output to a log file via log[tu]= option
 - ◆ t=truncate, u=unique (w/pid)
 - ◆ logs dt's command line at top of log
- Log files are opened in append mode (historic)
- **Beware:** No mounted file system checks!
- Tru64 Unix & Solaris, skip first 8K to avoid overwriting the disk label (e.g. position=8k)
- Success exit status is zero (0)
- General failure exit status is 255
- End of file/media exit status is 254
- Signals cause non-zero exit status too

DT Examples

- For file system I/O tests:
 - # dt of=/var/tmp/dt.data min=b max=256k incr=var oflags=trunc limit=1g \
pattern=iot iotype=random procs=10
 - OR
 - # dt of=/var/tmp/dt.data min=1 max=256k incr=var oflags=trunc limit=1g \
enable=lbdata files=10 sdirs=3 depth=10 procs=2
 - ◆ Alternate between buffered I/O and direct I/O via flags=direct
- For raw I/O disk tests:
 - # dt of=/dev/rhdisk7 min=b max=256k incr=var pattern=iot slices=10 aios=4
- For performance disk tests:
 - # dt of=/var/tmp/dt.data bs=64k limit=1g disable=compare,fsync
 - OR
 - # dt of=/dev/rhdisk7 bs=64k aios=32 disable=compare
- It's advisable to alternate between sequential and random I/O, and when doing sequential, alternate between forward and reverse directions.
- See [Test Recommendations](#) and [Use Cases](#) for more details.

Large Capacity Disk Testing

- **TEST DESCRIPTION:** This test shows a method to verify large TB sized disks without doing I/O to the full capacity. 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=4* to queue multiple requests per process for improved performance.
 - ◆ *step=4g* to seek after every record, thereby 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/rhdisk10 \
          slices=16 \
          step=4g \
          disable=pstats \
          aios=4 \
          oncerr=abort \
          min=b \
          max=256k \
          incr=var \
          align=rotate \
          pattern=iot \
          iodir=reverse \
          prefix='%d@%h' \
          alarm=5s \
          noprogt=10s
```

Large File Testing

- **TEST DESCRIPTION:** Creates a single large file of 15TB, then starts 15 processes each doing I/O to their own slice. The step option is used to skip 25MB between records, writing approximately 153GB's of data.

```
# dt \
    of=${OutputFile} \
    aios=8 \
    flags=direct \
    limit=15t \
    step=25m \
    slices=15 \
    oncerr=abort \
    bs=256k \
    dispose=keep \
    enable=debug \
    logu=logs/dt.log
```

Note: The example requires dt version 16.14, otherwise add iodir=reverse

No Progress I/O Timing

- `alarm=value` and `noprogt=value` options useful to detect slow or no I/O times, esp. during failovers.
- `alarm()` API previously used, now replaced with monitoring thread to avoid issues with I/O blocking signal delivery.
- Example [no-progress trigger script](#).
- Also see [I/O Monitoring Tool \(iomon\) Wiki](#).

Example:

```
shaix11# ./dt of=/dev/rhdisk7 bs=64k aios=16 pattern=iot \  
alarm=3s noprogt=15s runtime=5m stats=brief  
[ Initiate controller takeover to force path failover. ]  
dt: No progress made during write on /dev/rhdisk7 for 18 seconds!  
dt: No progress made during write on /dev/rhdisk7 for 21 seconds!  
...  
dt: No progress made during write on /dev/rhdisk7 for 39 seconds!
```

Copy and Verify I/O Mode

- *dt*'s `iomode=` option controls {copy, test, or verify} modes.
- The copy is an image mode copy, automatically verified.
- Useful for copying CD/DVD ISO's to disk (preferred over *dd*).

Example:

```
shaix11# dt of=/dev/rhdisk0 bs=64k pattern=iot aios=16 stats=none (prime hdisk0 data)
shaix11# dt if=/dev/rhdisk0 of=/dev/rhdisk1 bs=64k aios=16 iomode=copy
```

Copy Statistics:

```
Data operation performed: Copied '/dev/rhdisk0' to '/dev/rhdisk1'.
Total records processed: 16384 @ 65536 bytes/record (64.000 Kbytes)
Total bytes transferred: 1073741824 (1048576.000 Kbytes, 1024.000 Mbytes)
Average transfer rates: 35078139 bytes/sec, 34255.995 Kbytes/sec
Number I/O's per second: 535.250
Total passes completed: 0/1
Total errors detected: 0/1
Total elapsed time: 00m30.61s
Total system time: 00m00.79s
Total user time: 00m00.06s
Starting time: Wed May 16 09:53:02 2007
Ending time: Wed May 16 09:53:33 2007
```

```
...
shaix11# dt if=/dev/rhdisk1 bs=64k aios=16 pattern=iot stats=brief (verify hdisk1 data)
/dev/rhdisk1 Totals: 2097152 blocks, 1024.000 Mbytes, errors 0/1, passes 1/1, elapsed 01m13.10s
shaix11#
```

Data Corruptions

- Data Corruption Example (forced):

```
# dt of=dt.data count=3 dispose=keep disable=stats bs=64k pattern=iot
# dt if=dt.data count=3 disable=stats bs=64k enable=lbddata
dt: Error number 1 occurred on Wed Jul 29 11:41:03 2009
dt: Elapsed time since beginning of pass: 00m00.00s
dt: Elapsed time since beginning of test: 00m00.00s
dt: Data compare error at byte 4 in record number 1
dt: Relative block number where the error occurred is 0, position 4 (offset 4)
dt: Data expected = 0x39, data found = 0x1, byte count = 65536
dt: The correct data starts at address 0x529030 (marked by asterisk '*')
dt: Dumping Pattern Buffer (base = 0x529030, offset = 0, limit = 4 bytes):
Offset
000000 *39 9c c3 39

dt: The incorrect data starts at address 0x52e004 (marked by asterisk '*')
dt: Dumping Data Buffer (base = 0x52e000, offset = 4, limit = 512 bytes):
Offset
000000 00 00 00 00*01 01 01 01 02 02 02 02 03 03 03 03
000016 04 04 04 04 05 05 05 05 06 06 06 06 07 07 07 07
...
000480 78 78 78 78 79 79 79 79 7a 7a 7a 7a 7b 7b 7b 7b
000496 7c 7c 7c 7c 7d 7d 7d 7d 7e 7e 7e 7e 7f 7f 7f 7f

dt: Rereading and verifying record data using Direct I/O...
dt: Searched to block 0 (0) at byte position 0
dt: Record #1 - Reading 65536 bytes (128 blocks) into buffer 0x53f000, lba's 0 - 127 (pos 0)
dt: Reread data matches previous data read, possible write failure!
#
```

False Data Corruptions?

- **Beware:** False data corruptions *will* occur doing random I/O with unique data.
 - ◆ Unique data means different pattern data in each block.
- **Why?** Random I/O oftentimes overwrites previously written data block(s).
 - ◆ Resolving this requires maintaining a bitmap of blocks written to prevent overwrites.
- **Workaround:** Don't use pattern files larger than the device block size (512, 1K, etc).

Example:

```
shaix11# dt of=/var/tmp/dt.data min=b max=256k pf=pattern_all limit=100m iotype=random dlimit=32 stats=brief
End of Write pass 0/1, 204800 blocks, 100.000 Mbytes, 895 records, errors 0/1, elapsed 00m04.05s
```

```
dt: Error number 1 occurred on Thu May 17 10:10:03 2007
dt: Elapsed time since beginning of pass: 00m00.00s
dt: Elapsed time since beginning of test: 00m04.05s
dt: Data compare error at byte 0 in record number 4
dt: Relative block number where the error occurred is 142297, position 72856064
dt: Data expected = 0xa1, data found = 0x9d, byte count = 2048
dt: The correct data starts at address 0x20006c00 (marked by asterisk '*')
dt: Dumping Pattern Buffer (base = 0x20006000, offset = 3072, limit = 32 bytes):
```

```
0x20006bf0 f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff
0x20006c00 *a1 38 61 a2 1e 0f a4 bd 0a 3d 6d f0 ac c3 ac b4
```

```
dt: The incorrect data starts at address 0x2000c000 (marked by asterisk '*')
dt: Dumping Data Buffer (base = 0x2000c000, offset = 0, limit = 32 bytes):
```

```
0x2000c000 *9d 9d 86 d2 aa 82 67 b7 d3 a6 13 cd 41 6c 7d f4
0x2000c010 36 c7 ba 04 73 9d 86 16 a5 e4 57 a6 02 ed da 71
```

```
End of Read pass 1/1, 10 blocks, 0.005 Mbytes, 4 records, errors 1/1, elapsed 00m00.00s
/var/tmp/dt.data Totals: 204810 blocks, 100.005 Mbytes, errors 1/1, passes 1/1, elapsed 00m04.05s
shaix11#
```

Limitations/Notes

- One device per process in test mode.
- AIO used in lieu of threads to increase I/O's.
- AIO read-after-write (enable=raw) option does synchronous read(), thus slowing down AIO's.
- Encoded LBA's wrap at 2TB with 512 byte block size.
- Missing file system tests:
 - ◆ file locking
 - ◆ append/truncate (enabled via oflags=)
 - ◆ holes (read back as zeros, step= creates)
 - ◆ automatic open() file flags (randomly select)
 - ◆ more directory operations (rename, etc)
 - ◆ etc.

Remember: *dt* wasn't designed for file systems!

Documentation?

- “dt help” gives option summary.
- DT Overview ([dt-Overview.ppt](#))
- DT User’s Guide ([dt-UsersGuide.doc](#))
- DT Source Overview ([DtSourceOverview.doc](#))
- Above docs contained in DT source kit.
- Also available off my [DT web page](#).

Where is DT Located?

- Within NetApp:
<http://web.rtp.netapp.com/~rtmiller/dt.html>
- Latest Executables:
/u/rtmiller/Tools/dt.d-WIP/{OSdir}
- Perforce Source:
 - ◆ //depot/tools/main/performance/
 - ◆ subdirectory dt.d-v15.27/ (and later)
- Contact author via e-mail to:
Robin.Miller@NetApp.com
- Questions and comments welcome!