DTrace Quick Reference

DTrace Providers

Provider	Description
`dtrace`	Provides several probes related to DTrace itself. Use these probes to initialize state before tracing begins, process state after tracing has completed, and handle unexpected execution errors in other probes.
`lockstat`	Provides probes that can be used to discern lock contention statistics, or to understand virtually any aspect of locking behavior.
`profile`	Provides probes associated with a time-based interrupt firing every fixed, specified time interval.
`fbt`	Provides probes associated with the entry to and return from most functions in the Solaris kernel.
`syscall`	Provides a probe at the entry to and return from every system call in the system.
`sdt`	Provides probes at sites that a programmer has formally designated. This provider allows programmers to choose locations of interest to DTrace users and convey some knowledge about each location through the probe name.
`sysinfo`	Provides probes that correspond to kernel statistics classified by the name `sys`.
`vminfo`	Provides probes that correspond to the `vm` kernel statistics.
`proc`	Provides probes pertaining to the following activities: process creation and termination, LWP creation and termination, execution of new program images, and sending and handling signals.
`sched`	Provides probes related to CPU scheduling. Because a CPU is the one resource that all threads must consume, the `sched` provider is very useful for understanding systemic behavior.
`io`	Provides probes related to disk input and output.
`mib`	Provides probes that correspond to counters in the Solaris management information bases (MIBs).
`fpuinfo`	Provides probes that correspond to the simulation of floating-point instructions on SPARC microprocessors.
`pid`	Allows for tracing of the entry and return of any function in a user process as well as any instruction as specified by an absolute address or function offset.
`plockstat`	Provides probes that can be used to observe the behavior of user-level synchronization primitives, including lock contention and hold times.
`fasttrap`	Allows for tracing at specific, preprogrammed user process locations.

DTrace Functions

Name	Prototype	Description
`trace`	`void trace (expression)`	Takes a D expression as argument and traces the result to the directed buffer.
`tracemem`	`void tracemem (address, size_t nbytes)`	Takes the memory address specified by `address` into the directed buffer for the length specified by `nbytes`. Address is a D expression.
`printf`	`void printf (string format,...)`	The arguments are a format string followed by a variable number of arguments. The arguments are formatted for output according to the specified format string.
`printa`	`void printa (aggregation)` `void printa (string`	Enables displaying and formatting of aggregations. If a format is not specified, the default format is used.

	format, aggregation)	
stack	void stack (int nframes), void stack (void)	Records a kernel stack trace, nframes in depth. The number specified by the stackframes option is used if nframes is not specified. May also be used as a key to an aggregation.
ustack	void ustack (int nframes, int strsize) void ustack (int nframes) void ustack (void)	Records a user stack trace, nframes in depth. The number specified by the ustackframes option is used if nframes is unspecified. If strsize is specified and non-zero, ustack() will allocate the specified amount of string space and use it to perform address-to-symbol translation directly from the kernel.
jstack	void jstack (int nframes, int strsize) void jstack (int nframes) void jstack (void)	Alias for ustack() that uses the jstackframes option for the stack frame value and jstackstrsize for the string space size.
stop	void stop (void)	Forces the process that fires the enabled probe to stop when it next leaves the kernel.
raise	void raise (int signal)	Sends the specified signal to the currently running process.
copyout	void copyout (void *buf, uintptr_t addr, size_t nbytes)	Copies nbytes from the buffer buf to the address addr in the address space of the process associated with the current thread.
copyoutstr	void copyoutstr (string str, uintptr_t addr, size_t maxlen)	Copies the string str to the address addr in the address space of the process associated with the current thread. The string length is limited to the value set by the strsize option.
system	void system (string program, ...)	Causes program to be executed as if it were given to the shell as input. Program may contain any of the printf/printa formats. Other arguments must match the specified format in program.
breakpoint	void breakpoint (void)	Induces a kernel breakpoint, causing the system to stop and transfer control to the kernel debugger.
panic	void panic (void)	Causes a kernel panic. Should be used to force a system crash dump at a time of interest.
chill	void chill (int nanoseconds)	Causes DTrace to spin for the given nanoseconds. For system safety, DTrace will refuse to execute the chill action for more than 500 milliseconds in each 1-second on any CPU.
exit	void exit (int status)	Immediately stops tracing, notifies DTrace consumer to cease tracing, performs any final processing, and calls exit with the specified status.
alloca	void *alloca (size_t size)	Allocates size bytes out of scratch space and returns a pointer to the allocated memory.
basename	string basename (char *str)	Creates a string that consists of a copy of the specified string, but without any prefix that ends in /.
bcopy	void bcopy (void *src, void *dest,	Copies size bytes from the memory pointed to by src, to the memory pointed to by dest. All source memory must

	size_t size)	lie outside of scratch memory, and all destination memory must lie within it.
cleanpath	string cleanpath (char *str)	Creates a string that consists of a copy of the path indicated by str, but with redundant elements eliminated. This might result in shorter invalid paths being returned.
copyin	void *copyin (uintptr_t addr, size_t size)	Copies the specified size in bytes from the specified user address into a DTrace scratch buffer and returns the address of this buffer. The resulting buffer pointer is 8-byte aligned.
copyinstr	string copyinstr (uintptr_t addr)	Copies a null-terminated C string from the specified user address into a DTrace scratch buffer, and returns the address of this buffer. The strsize option limits the string length.
copyinto	void copyinto (uintptr_t addr, size_t size, void *dest)	Copies the specified size in bytes from the specified user address into the DTrace scratch buffer specified by dest.
dirname	string dirname (char *str)	Creates a string that consists of all but the last level of the path name specified by str.
msgdsize	size_t msgdsize (mblk_t *mp)	Returns the number of bytes in the data message pointed to by mp.
msgsize	size_t msgsize (mblk_t *mp)	Returns the number of bytes in the message pointed to by mp.
mutex_owned	int mutex_owned (kmutex_t *mutex)	Returns non-zero if the calling thread currently holds the specified kernel mutex, or zero if the specified adaptive mutex is currently unowned.
mutex_owner	kthread_t *mutex_owner (kmutex_t *mutex)	Returns the thread pointer of the current owner of the specified adaptive kernel mutex. Returns NULL if the specified adaptive mutex is currently unowned, or if the specified mutex is a spin mutex.
mutex_type_adaptive	int mutex_type_adaptive (kmutex_t *mutex)	Returns non-zero if the specified kernel mutex is of type MUTEX_ADAPTIVE, or zero if it is not.
progenyof	int progenyof (pid_t pid)	Returns non-zero if the calling process is among the progeny of the specified process ID.
rand	int rand (void)	Returns a pseudo-random integer.
rw_iswriter	int rw_iswriter(krwlock_t *rwlock)	Returns non-zero if the specified reader-writer lock is either held or desired by a writer. Returns zero if the lock is held only by readers, no writer is blocked, or the lock is not held at all.
rw_write_held	int rw_write_held (krwlock_t *rwlock)	Returns non-zero if the specified reader-writer lock is currently held by a writer. Returns zero if the lock is held only by readers or not held at all.
speculation	int speculation (void)	Reserves a speculative trace buffer for use with speculate() and returns an identifier for this buffer.
strjoin	string strjoin(char *str1 char *str2)	Creates a string that consists of str1 concatenated with str2.
strlen	size_t strlen(string str)	Returns the length of the specified string in bytes, excluding the terminating null byte.

DTrace Aggregating Functions

Name	Arguments	Result
`count`	none	Number of times called.
`sum`	scalar expression	Total value of the specified expressions.
`avg`	scalar expression	Arithmetic average of the specified expressions.
`min`	scalar expression	Smallest value among the specified expressions.
`max`	scalar expression	Largest value among the specified expressions.
`lquantize`	scalar expression, lower bound, upper bound, step value	A linear frequency distribution, sized by the specified range, of the values of the specified expressions. Increments the value in the **highest** bucket that is **less** than the specified expression.
`quantize`	scalar expression	A power-of-two frequency distribution of the values of the specified expressions. Increments the value in the **highest** power-of-two bucket that is **less** than the specified expression.

DTrace Variables

Variable	Description	Usage
Scalar	Used to represent fixed-size data objects. They could be individual such as pointers and integers or composite such as arrays.	`x=123` (scalar variable *x* of type integer)
Associative array	Used to represent collections of data elements that can be retrieved by specifying a key. There is no predefined limit of the number of elements. Elements can be indexed by any tuple, and elements are not stored in preallocated consecutive storage locations.	`a[123,"hello"] = 56` (associative array *a* with key, *[int, string]*)
Thread-Local	Used to declare variable storage that is local to each operating system thread.	`self->x= 45` (thread-local variable *x* of type integer)
Clause-Local	This variable is active for the lifetime of a given probe clause and its storage is reused for each D program clause.	`this->c='D'` (character clause-local variable *c*)
Built-in	All these variables are scalar global variables.	--
External	The backquote character (`) is a scoping operator for accessing variables that are defined in the OS and not in your D program.	`` `kmem_flags `` (accessing a C variable in the kernel source code)

DTrace Built-in Variables

Type and Name	Description
`int64_t arg0, ..., arg9`	The first 10 input arguments to a probe represented as raw 64-bit integers. If fewer than 10 arguments are passed to the current probe, the remaining variables return zero.
`args[]`	The typed arguments to the current probe, if any. `args[]` array is accessed using an integer index, but each element is defined to be the type corresponding to the given probe argument.
`uintptr_t caller`	The program counter location of the current thread just before entering the current probe.
`chipid_t chip`	The CPU chip identifier for the current physical chip.
`processorid_t cpu`	The CPU identifier for the current CPU.
`cpuinfo_t *curcpu`	The CPU information for the current CPU.
`lwpsinfo_t *curlwpsinfo`	The lightweight process (LWP) state of the LWP associated with the current thread.
`psinfo_t *curpsinfo`	The process state of the process associated with the current thread.
`kthread_t`	The address of the operating system kernel's internal data structure for the current

`*curthread`	thread, the `kthread_t`. `kthread_t` is defined in `<sys/thread.h>`.
`string cwd`	The name of the current working directory of the process associated with the current thread.
`uint_t epid`	The enabled probe ID (EPID) for the current probe. This integer uniquely identifies a particular probe that is enabled with a specific predicate and set of actions.
`int errno`	The error value returned by the last system call executed by this thread.
`string execname`	The name that was passed to `exec`(2) to execute the current process.
`gid_t gid`	The real group ID of the current process.
`uint_t id`	The probe ID for the current probe. This ID is the system-wide unique identifier for the probe as published by DTrace.
`uint_t ipl`	The interrupt priority level (IPL) on the current CPU at probe firing time.
`lgrp_id_t lgrp`	The latency group ID for the latency group of which the current CPU is a member.
`pid_t pid`	The process ID of the current process.
`pid_t ppid`	The parent process ID of the current process.
`string probefunc`	The function name portion of the current probe's description.
`string probemod`	The module name portion of the current probe's description.
`string probename`	The name portion of the current probe's description.
`string probeprov`	The provider name portion of the current probe's description.
`psetid_t pset`	The processor set ID for the processor set containing the current CPU.
`string root`	The name of the root directory of the process associated with the current thread.
`uint_t stackdepth`	The current thread's stack frame depth at probe firing time.
`id_t tid`	The thread ID of the current thread. For threads associated with user processes, this value is equal to the result of a call to `pthread_self`.
`uint64_t timestamp`	The current value of a nanosecond timestamp counter. This counter increments from an arbitrary point in the past and should only be used for relative computations.
`uid_t uid`	The real user ID of the current process.
`uint64_t uregs[]`	The current thread's saved user-mode register values at probe firing time.
`uint64_t vtimestamp`	The current value of a nanosecond timestamp counter that is the amount of time the current thread has been running on a CPU, minus the time spent in DTrace predicates and actions.
`uint64_t walltimestamp`	The current number of nanoseconds since 00:00 Universal Coordinated Time, January 1, 1970.