The DTrace Cheatsheet

`dtrace -n 'probe /predicate/ { action; }'`

http://wikis.sun.com/display/DTrace/Documentation

FINDING PROBES

`dtrace -l | grep foo`	keyword search
`dtrace -n 'fbt:::entry { @[probefunc] = count(); }' -c 'ping host'`	frequency count
`grep foo /opt/DTT/Bin/*`	DTraceToolkit

PROBE ARGUMENTS

`syscall:::`	`man syscallname`
`fbt:::`	Kernel source

PROBES

`BEGIN`	program start
`END`	program end
`tick-1sec`	run once per sec, one CPU only
`syscall::read*:entry`	process reads
`syscall::write*:entry`	process writes
`syscall::open*:entry`	file open
`proc:::exec-success`	process create
`io:::start,io:::done`	disk or NFS I/O request
`lockstat:::adaptive-block`	blocked thread acquired kernel mutex
`sysinfo:::xcalls`	cross calls
`sched:::off-cpu`	thread leaves CPU
`fbt:::entry`	entire kernel
`profile-123`	sample at 123Hz
`perl*:::`	Perl provider probes
`javascript*:::`	JavaScript provider probes

VARS

`execname`	on-CPU process name
`pid, tid`	on-CPU PID, Thread ID
`cpu`	CPU ID
`timestamp`	time, nanoseconds
`vtimestamp`	time thread was on-CPU, ns
`arg0...N`	probe args (uint64)
`args[0]...[N]`	probe args (typed)
`curthread`	pointer to current thread
`probemod`	module name
`probefunc`	function name
`probename`	probe name
`self->foo`	thread-local
`this->foo`	clause-local
`$1...$N`	CLI args, int
`$$1...$$N`	CLI args, str
`$target`	`-p PID`, `-c command`
`curpsinfo`	procfs style process statistics

ACTIONS

`@agg[key1, key2] = count()`	frequency count
`@agg[key1, key2] = sum(var)`	sum variable
`@agg[key1, key2] = quantize(var)`	power of 2 quantize variable
`printf("format", var0...varN)`	print vars; use `printa()` for aggrs
`stack(num), ustack(num)`	print num lines of kernel, user stack
`func(pc), ufunc(pc)`	return kernel, user func name from PC
`clear(@)`	clear an aggregation
`trunc(@, 5)`	truncate agg to top 5 entries
`stringof(ptr)`	string from kernel address
`copyinstr(ptr)`	string from user address
`exit(0);`	exit dtrace

SWITCHES

`-n`	trace this probe description
`-l`	list probes instead of tracing them
`-q`	quiet; don't print default output
`-s <file>`	invoke script file, or at top of script: `#!/usr/sbin/dtrace -s`
`-w`	allow destructive actions
`-p PID`	allow `pid:::` provider to trace this pid; it's also `$target`
`-c 'command'`	have dtrace invoke this command
`-o file`	output to file
`-x options`	set various DTrace options (`switchrate`, `bufsize`...)

PRAGMAS

`#pragma D option quiet`	same as `-q`, quiet output
`#pragma D option destructive`	same as `-w`, allow destructive actions
`#pragma D option switchrate=10hz`	print at 10Hz (instead of 1Hz)
`#pragma D option bufsize=16m`	set per-CPU buffer size (default 4m)
`#pragma D option defaultargs`	$1 is 0, $$1 is "", etc...

ONE-LINERS

```
dtrace -n 'proc:::exec-success { trace(curpsinfo->pr_psargs); }'
dtrace -n 'syscall:::entry { @num[execname] = count(); }'
dtrace -n 'syscall::open*:entry { printf("%s %s", execname, copyinstr(arg0)); }'
dtrace -n 'io:::start { @size = quantize(args[0]->b_bcount); }'
dtrace -n 'fbt:::entry { @calls[probemod] = count(); }'
dtrace -n 'sysinfo:::xcalls { @num[execname] = count(); }'
dtrace -n 'profile-1001 { @[stack()] = count() }'
dtrace -n 'profile-101 /pid == $target/ { @[ustack()] = count() }' -p PID
dtrace -n 'syscall:::entry { @num[probefunc] = count(); }'
dtrace -n 'syscall::read*:entry { @[fds[arg0].fi_pathname] = count(); }'
dtrace -n 'vminfo:::as_fault { @mem[execname] = sum(arg0); }'
dtrace -n 'sched:::off-cpu /pid == $target/ { @[stack()] = count(); }' -p PID
dtrace -n 'pid$target:libfoo::entry { @[probefunc] = count(); }' -p PID
```