

Aug 06, 05 15:28	dtrace.cookbook	Page 1/3
Cookbook: DTrace		
Wichtige Kommandos:		
<code>dtrace # CLI-Zugang zu den DTrace Features; siehe dtrace(1M)</code>		
Nutzung:		
<code>dtrace -n '... DTrace Script ...'</code>		
Laengere Scripts: Datei mit folgendem Inhalt schreiben		
<pre>#!/usr/sbin/dtrace -s ... DTrace Script ...</pre>		
Dtrace Script: besteht aus ein oder mehreren Clauses		
<code><probe> <predicate> <action></code>		
<code><probe> ::= module:provider:function:name</code> # Teile koennen fehlen		
<code><predicate> ::= / ... Bedingung ... /</code> # mit Variablen (eigene / vordefinierte)		
<code><action> ::= { ... dtrace Statements ... }</code>		
<code><name></code> ist der Messpunkt in der Funktion, meist <code>entry</code> (Start der Funktion) oder <code>return</code> (Ende)		
Beispiele:		
1. Alle Probes im Betriebssystem:		
<code>dtrace -l</code>		
2. Anzeige aller Systemcalls im System (Vorsicht auf Produktionssystemen!)		
<pre>dtrace -n ' syscall:::entry { trace(execname) }' # # Probe: - Nur Systemcalls # - Alle Module # - Alle Funktionen # - Nur Funktions-Starts # # Predicate: keine Einschraenkung # # Action: - trace (Ausgabe) des Funktionsnamens, # (execname enthaelt den Namen des Programmes) # - dazu wird der angesteuerte Probe ausgegeben. # (Ausschalten mit -q)</pre>		
dtrace kann mit Control-C gestoppt werden.		
3. Anzeige der Systemcalls des Kommandos ls		

Aug 06, 05 15:28	dtrace.cookbook	Page 2/3
<pre>dtrace -n ' syscall:::entry / execname == "ls" / { trace(probefunc) }' # # wie oben, Einschraenkung auf das Kommando ls # execname enthaelt immer den Namen des Prozesses</pre>		
Das dtrace-Kommando in einem Fenster ausfuehren. In einem anderen Fenster ein ls-Kommando ausfuehren.		
dtrace dann wieder mit Control-C stoppen.		
4.a Statistik ueber den Systemcall read		
Zum Beispiel hat man im iostat gesehen, das ein paar Platten viele <code>read()</code> Systemcalls haben, und man moechte herausfinden, welche Applikation dieses verursacht. Das geht mit:		
<pre>dtrace -n 'syscall:::read:entry { @[execname] = count() }' # # Die Probe feuert bei jedem Systemcall read. # @s ist eine Aggregation, in der dann nach Programmname getrennt # hochgezahlt wird.</pre>		
Eine Zeitlang zaehlen lassen und dann dtrace mit Control-C abbrechen. Dann wird die Statistik ausgegeben.		
4.b Statistik ueber den Systemcall read mit automatischer Ausgabe		
Zum vorigen dtrace-script nach <code>}</code> und vor <code>'</code> das folgende hinzufuegen:		
<pre>tick-10sec { printa(@s); clear(@s) } # # Der Probe tick-10sec wird alle 10 Sekunden gefeuert. # printa() gibt die Aggregation aus, so wie sie bisher steht. # clear() loescht die Felder der Aggregation</pre>		
So kann man alle dtrace-Statistiken in ein Status-Kommando wie <code>vmstat</code> , <code>iostat</code> ... verwandeln		
5.a Statistik ueber die Groesse der read		
<pre>dtrace -n 'syscall:::read:entry { @[execname] = quantize(arg2) }'</pre>		
In einem Fenster starten und in einem anderen Fenster das Kommando		
<code>dd if=/dev/zero of=/dev/null bs=512k count=137</code>		
laufen lassen. Dann das dtrace-Kommando mit Control-C abbrechen. Man erhaelt die Statistik ueber die Groesse der Puffer, die bei <code>read</code> spezifiziert worden sind. (<code>arg2</code> ist der 3. Parameter von <code>read</code> , siehe: <code>man -s2 read</code>)		
5.b Statistik ueber die tatsaechlich gelesenen Zeichen		
Die Anzahl der gelesenen Zeichen weiss man erst beim Ende des Systemcalls. Die Anzahl wird als Return-Wert beim <code>read</code> -Systemcall zurueckgeliefert. Dieser Wert ist bei dem <code>read:return</code> Probe in der Variable <code>arg0</code> verfuegbar (siehe dtrace-Handbuch).		

```
dtrace -n 'syscall::read:return { @[execname] = quantize(arg0) }'
```

Last erzeugen wieder mit

```
dd if=/dev/zero of=/dev/null bs=512k count=251
```

Und Abbruch des dtrace mit Control-C.

(-1 heisst: kein Zeichen wurde gelesen; siehe: man -s2 read)

6. Ausgabe der Dateinamen, die geöffnet werden.

Der beim Öffnen von Dateien wird der Dateiname als 1. Parameter uebergeben.

Den Dateinamen muss man vor der Ausgabe in den System-Bereich holen, da ja die Datensammlung von dtrace im Kernel abläuft.

Einige Variablen:

```
pid           # Prozess-ID
ppid          # Prozess-ID des Parent Prozesses
probefunc     # Name der Funktion
execname      # Name des Programmes
zonename      # Name der Zone, in der der Prozess laeuft
arg0, arg1, ... # bei :entry : Parameter der Funktion
arg0          # bei :return : Returnwert der Funktion
```