NAME

dwatch - watch processes as they trigger a particular DTrace probe

SYNOPSIS

dwatch [**-1defFmnPqRvVwxy**] [**-B** *num*] [**-E** *code*] [**-g** *group*] [**-j** *jail*] [**-k** *name*] [**-K** *num*] [**-N** *count*] [**-o** *file*] [**-O** *cmd*] [**-p** *pid*] [**-r** *regex*] [**-t** *test*] [**-T** *time*] [**-u** *user*] [**-X** *profile*] [**-z** *regex*] [**--**] [probe[,...]] [args ...]
dwatch -l [**-fmnPqy**] [**-r** *regex*] [probe ...]
dwatch -Q [**-1qy**] [**-r** *regex*]

DESCRIPTION

The **dwatch** utility uses dtrace(1) to display process info when a given DTrace probe point is triggered. Only the root user or users with sudo(8) (*ports/security/sudo*) access can run this command.

dwatch automates the process of generating DTrace scripts to coalesce trace output by date/time, process info, and [optionally] probe-specific data.

Output format without options is:

date/time uid.gid execname[pid]: psargs

For example, the command ‘dwatch BEGIN’ produces:

INFO Watching ’dtrace:::BEGIN’ ...
2017 May 29 08:23:20 0.0 dtrace[60671]: dtrace -s /dev/stdin

The **-F** option causes **dwatch** to instead coalesce trace output by date/time, process info, and probe traversal.

Output format with the ‘**-F**’ option is:

date/time uid.gid execname[pid]: {->,<-, |} prov:mod:func:name ...

For example, the command ‘dwatch -F BEGIN’ produces:

INFO Watching ’dtrace:::BEGIN’ ...
2017 May 29 21:34:41 0.0 dtrace[86593]: | dtrace:::BEGIN ...

The **-R** option causes **dwatch** to display a process tree containing the parent, grandparent, and ancestor process info.

Output format with the '**-R**' option is:

```
date/time uid0.gid0 execname[pid0]: psargs0
 -+= pid3 uid3.gid3 psargs3
  \-+= pid2 uid2.gid2 psargs2
    \-+= pid1 uid1.gid1 psargs1
      \-+= pid0 uid0.guid0 psargs0
```

For example, the command 'dwatch -R BEGIN' produces:

```
INFO Watching 'dtrace:::BEGIN' ...
2017 May 29 21:38:54 0.0 dtrace[86899]: dtrace -s /dev/stdin
 -+= 86855 604.604 -bash
  \-+= 86857 604.604 /bin/sh /usr/sbin/dwatch -R BEGIN
    \-+= 86897 0.0 sudo dtrace -s /dev/stdin
      \-+= 86899 0.0 dtrace -s /dev/stdin
```

Of particular interest is the ability to filter using regular expressions. The '**-g** *group*', '**-p** *pid*', '**-r** *regex*', '**-u** *user*', and '**-z** *regex*' options can be combined with '**-R**' to match on parent process criteria as well as current process info.

In contrast, the '**-j** *jail*', and '**-k** *name*' options apply only to the current process even if '**-R**' is given.

The '**-E** *code*' option gives the ability to customize probe-specific data. For example, the command:

```
dwatch -E 'printf("%s", copyinstr(arg0))' chdir
```

displays the path argument sent to chdir(2) calls.

Profiles can be written for more complex routines and/or convenience. To list available profiles use the '**-Q**' option. Use the '**-X** *profile*' option to use a particular profile.

For example, the command 'dwatch -X kill' displays arguments sent to kill(2).

OPTIONS

If a *probe* argument does not contain colon (":") and none of '**-P**', '**-m**', '**-f**', or '**-n**' are given, the probe argument is intelligently mapped to its most-likely value. Use '**dwatch -l** *name*' to see what probes will match a given name.

Multiple probes must be given as a single (quoted) argument, separated by comma and/or whitespace.

Any/all arguments following said probes will be passed to dtrace(1) unmodified.

-1 Print one line per process/profile (Default; disables '**-R**').

-B *num* Maximum number of arguments to display (Default 64).

-d Debug. Send dtrace(1) script to stdout instead of executing.

-e Exit after compiling request but prior to enabling probes.

-E *code* DTrace *code* for event details. If '-', read from stdin. This allows customization of what is printed after date/time and process info. By default, the name and arguments of the program triggering the probe are shown. Can be specified multiple times.

-f Enable probes matching the specified function names.

-F Coalesce trace output by probe.

-g *group*
Group filter. Only show processes matching *group* name/gid. This can be an awk(1) regular expression to match a numerical gid.

-j *jail* Jail filter. Only show processes matching *jail* name/jid.

-k *name* Only show processes matching *name*. Can also be of the format 'name*' to indicate "begins with", '*name' to indicate "ends with", or '*name*' to indicate "contains". Can be specified multiple times.

-K *num* Maximum directory depth to display (Default 64).

-l List available probes on standard output and exit.

-m Enable probes matching the specified module names.

-X *profile*
Load profile from DWATCH_PROFILES_PATH.

-n Enable probes matching the specified probe names.

-N *count*

Exit after *count* matching entries (Default 0 for disabled).

-o *file* Set output file. If '-', the path '/dev/stdout' is used.

-O *cmd* Execute *cmd* for each event. This can be any valid sh(1) command. The environment variables '$TAG' and '$DETAILS' are set for the given *cmd*.

-p *pid* Process id filter. Only show processes with matching *pid*. This can be an awk(1) regular expression.

-P Enable probe matching the specified provider name.

-q Quiet. Hide informational messages and all dtrace(1) errors.

-Q List available profiles in DWATCH_PROFILES_PATH and exit.

-r *regex* Filter. Only show blocks matching awk(1) regular expression.

-R Show parent, grandparent, and ancestor of process.

-t *test* Test clause (predicate) to limit events (Default none). Can be specified multiple times.

-T *time* Timeout. The format is '#[smhd]' or just '#' for seconds.

-u *user* User filter. Only show processes matching *user* name/uid. This can be an awk(1) regular expression to match a numerical UID.

-v Verbose. Show all errors from dtrace(1).

-V Report **dwatch** version on standard output and exit.

-w Permit destructive actions (copyout*, stop, panic, etc.).

-x Trace. Print '<probe-id>' when a probe is triggered.

-y Always treat stdout as console (enable colors/columns/etc.).

-z *regex* Only show processes matching awk(1) regular expression.

PROFILES

Profiles customize the data printed during events. Profiles are loaded from a colon-separated list of directories in DWATCH_PROFILES_PATH. This is an incomplete list of profiles with basic descriptions:

chmod	Print mode and path from chmod(2), lchmod(2), fchmodat(2)
errno	Print non-zero errno results from system calls
io	Print disk I/O details provided by dtrace_io(4)
ip	Print IPv4 and IPv6 details provided by dtrace_ip(4)
kill	Print signal and pid from kill(2)
nanosleep	Print requested time from nanosleep(2)
open	Print path from open(2), openat(2)
proc	Print process execution details provided by dtrace_proc(4)
proc-signal	Print process signal details provided by dtrace_proc(4)
rw	Print buffer contents from read(2), write(2)
sched	Print CPU scheduling details provided by dtrace_sched(4)
tcp	Print TCP address/port details provided by dtrace_tcp(4)
tcp-io	Print TCP I/O details provided by dtrace_tcp(4)
udp	Print UDP I/O details provided by dtrace_udp(4)
vop_create	Print filesystem paths being created by VOP_CREATE(9)
vop_lookup	Print filesystem paths being looked-up by VOP_LOOKUP(9)
vop_mkdir	Print directory paths being created by VOP_MKDIR(9)
vop_mknod	Print device node paths being created by VOP_MKNOD(9)

vop_readdir Print directory paths being read by VOP_READDIR(9)

vop_remove
Print filesystem paths being removed by VOP_REMOVE(9)

vop_rename
Print filesystem paths being renamed by VOP_RENAME(9)

vop_rmdir Print directory paths being removed by VOP_RMDIR(9)

vop_symlink
Print symlink paths being created by VOP_SYMLINK(9)

ENVIRONMENT

These environment variables affect the execution of **dwatch**:

DWATCH_PROFILES_PATH If DWATCH_PROFILES_PATH is set, **dwatch** searches for profiles in the colon-separated list of directories in that variable instead of the default '/usr/libexec/dwatch:/usr/local/libexec/dwatch'. If set to NULL, profiles are not loaded.

EXIT STATUS

The **dwatch** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Watch processes entering system CPU scheduler.

```
dwatch on-cpu
```

List available profiles, one line per profile.

```
dwatch -1 -Q
```

Do not execute dtrace(1) but display script on stdout and exit.

```
dwatch -d fsync
```

Compile and test but do not execute code generated with given probe.

```
dwatch -e test_probe
```

Print argument one being passed to each call of zfs_sync().

```
dwatch -E ’printf("%i", arg1)’ zfs_sync
```

Watch all functions named ‘read’.

```
dwatch -f read
```

Watch all probe traversal.

```
dwatch -F :
```

Watch syscall probe traversal.

```
dwatch -F syscall
```

Display only processes belonging to wheel super-group.

```
dwatch -g wheel execve
```

Display only processes belonging to groups ‘daemon’ or ‘nobody’.

```
dwatch -g ’1|65534’ execve
```

Ignore jails, displaying only base system processes.

```
dwatch -j 0 execve
```

Display only processes running inside the jail named ‘myjail’.

```
dwatch -j myjail execve
```

Watch syscall traversal by ruby processes.

```
dwatch -k ’ruby*’ -F syscall
```

Watch syscall traversal by processes containing ‘daemon’ in their name.

```
dwatch -k ’*daemon*’ -F syscall
```

Watch signals being passed to kill(2).

```
dwatch -X kill
```

Watch signals being passed between bash(1) and vi(1).

```
dwatch -k bash -k vi -X kill
```

Display a list of unique functions available.

```
dwatch -l -f
```

List available probes for functions ending in ‘read’.

```
dwatch -l -f ’*read’
```

List available probes ending in "read".

```
dwatch -l -r ’read$’
```

Display a list of unique providers.

```
dwatch -l -P
```

Watch paths being removed by VOP_REMOVE(9).

```
dwatch -X vop_remove
```

Watch the name ‘read’ instead of the function ‘read’. The **dwatch** selection algorithm will commonly favor the function named ‘read’ when not given a type (using ‘**-P**’, ‘**-m**’, ‘**-f**’, or ‘**-n**’) because there are more probes matching the function named ‘read’ than probes matching ‘read’ for any other type.

```
dwatch -n read
```

Display the first process to call kill(2) and then exit.

```
dwatch -N 1 kill
```

Watch processes forked by pid 1234.

dwatch -p 1234 execve

Watch processes forked by either pid 1234 or pid 5678.

dwatch -p ’1234|5678’ execve

Watch the provider ‘random’ instead of the function ‘random’. The **dwatch** selection algorithm will commonly favor the function named ‘random’ when not given a type (using ‘**-P**’, ‘**-m**’, ‘**-f**’, or ‘**-n**’) because there are more probes matching the function named ‘random’ than probes matching the provider named ‘random’.

dwatch -P random

Display available profiles matching ‘vop’.

dwatch -Q -r vop

Watch VOP_LOOKUP(9) paths containing ‘/lib/’.

dwatch -r /lib/ -X vop_lookup

Show process tree for each command as it is executed.

dwatch -R execve

Watch processes forked by pid 1234 or children thereof.

dwatch -R -p 1234 execve

Display processes calling write(2) with "nbytes" less than 10.

dwatch -t ’arg2<10’ -E ’printf("%d",arg2)’ write

Display write(2) buffer when "execname" is not ‘dtrace’ and "nbytes" is less than 10.

dwatch -X write -t ’execname != "dtrace" && this->nbytes < 10’

Watch ‘statfs’ for 5 minutes and exit.

dwatch -T 5m statfs

Display only processes belonging to the root super-user.

```
dwatch -u root execve
```

Display only processes belonging to users ‘daemon’ or ‘nobody’.

```
dwatch -u ’1|65534’ execve
```

Print version and exit.

```
dwatch -V
```

View the first 100 scheduler preemptions.

```
dwatch -y -N 100 preempt | less -R
```

Display processes matching either "mkdir" or "rmdir".

```
dwatch -z ’(mk|rm)dir’ execve
```

Run a command and watch network activity only while that command runs.

```
dwatch -X tcp -- -c "nc -zvw10 google.com 22"
```

Watch open(2) and openat(2) calls only while pid 1234 is active.

```
dwatch -X open -- -p 1234
```

Watch probe traversal for a given command. Note that "-c true" is passed to dtrace(1) since it appears after the **dwatch** probe argument.

```
dwatch -F ’pid$target:::entry’ -c true
```

SEE ALSO

dtrace(1)

HISTORY

dwatch first appeared in FreeBSD 11.2.

AUTHORS

Devin Teske <*dteske@FreeBSD.org*>