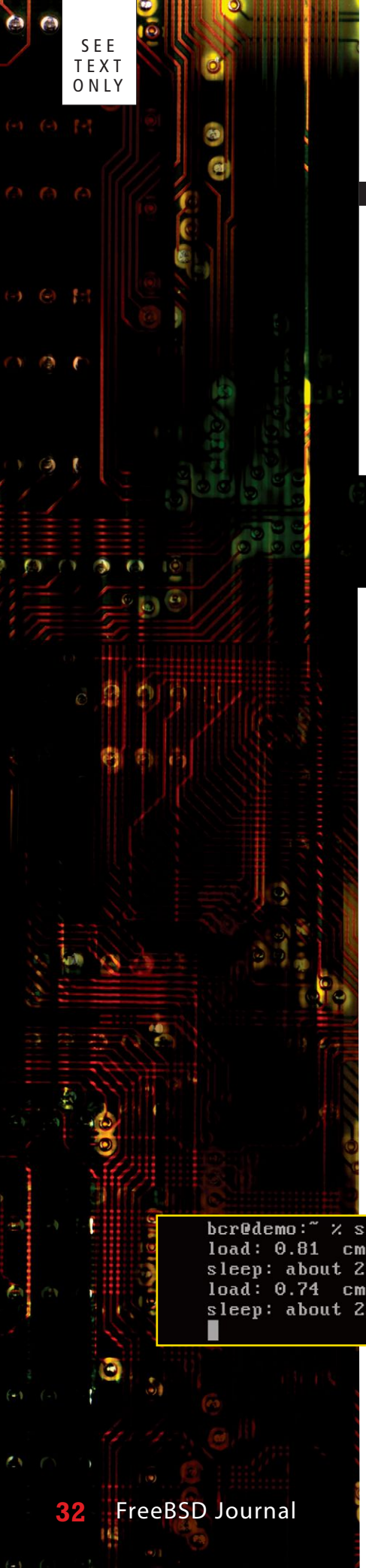


SEE TEXT ONLY

BY BENEDICT REUSCHLING

Linux, and a couple other Unix Systems that do not implement

SIGINFO

STAY IN THE DARK!

Every now and then in a sysadmin's daily activities, there comes a time when a command takes a little longer before it gives back the prompt. This may be due to many causes. Maybe there is a process that is either waiting for input from another source or is busy processing something and has nothing to print to the console in the meantime. The longer there is no output on the screen, the more nervous the user becomes and starts to wonder whether the process is still active or is progressing at all. In those situations, users logged into a BSD system are fortunate to have SIGINFO available. By pressing CTRL+T, many utilities print out additional information that would normally be left hidden.

As the name suggests, SIGINFO is a signal that can be sent to a process like CTRL+Z (SIGTSTP), to stop the process, or SIGKILL to terminate a hung process. Not every operating system implements SIGINFO, as it is not part of the POSIX standard. A look at signal(3) reveals that FreeBSD implements SIGINFO. Many sysadmins on those systems have come to rely on it for that extra bit of information. Linux and a couple other Unix systems that do not implement SIGINFO stay in the dark, as nothing will be returned no matter how often CTRL+T is pressed.

The clearest example to demonstrate this is probably the sleep(1) command. The following example shows the number of remaining seconds by hitting CTRL+T two times in short intervals:

```
bcr@demo:~ % sleep 30
load: 0.81  cmd: sleep 695 [nanslp] 2.08r 0.00u 0.00s 0% 1468k
sleep: about 27 second(s) left out of the original 30
load: 0.74  cmd: sleep 695 [nanslp] 7.09r 0.00u 0.00s 0% 1476k
sleep: about 22 second(s) left out of the original 30
```

The dd command is another example of a process that can run for a long time. In this case, we create the base disk image for a virtual machine and fill it from /dev/zero. SIGINFO returns a short report and shows how many records have already been written. (next page)

```
bcr@demo:~ % dd if=/dev/zero of=vm.img bs=1024 count=900000
load: 0.79  cmd: dd 705 [dmu_tx_delay] 1.35r 0.03u 0.60s 4% 1544k
127210+0 records in
127210+0 records out
130263040 bytes transferred in 1.350269 secs (96471888 bytes/sec)
load: 0.80  cmd: dd 705 [runnable] 2.93r 0.04u 1.26s 8% 1548k
256204+0 records in
256204+0 records out
262352896 bytes transferred in 2.934288 secs (89409377 bytes/sec)
```

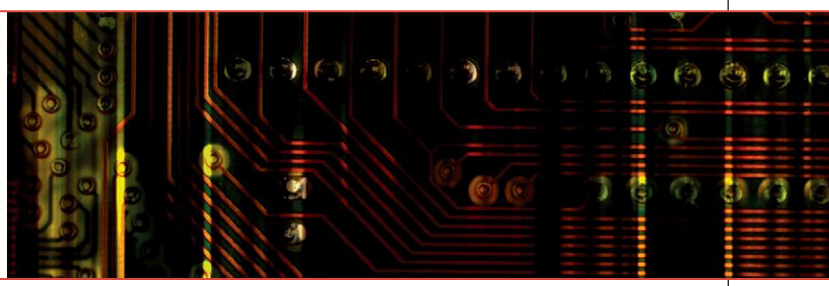
Network utilities like ping(8) and tcpdump(1) will list the received or captured packets respectively.

```
bcr@demo:~ % ping google.com
PING google.com (172.217.22.46): 56 data bytes
64 bytes from 172.217.22.46: icmp_seq=0 ttl=57 time=6.619 ms
load: 0.86  cmd: ping 710 [select] 0.83r 0.00u 0.00s 0% 1836k
1/1 packets received (100.0%) 6.619 min / 6.619 avg / 6.619 max
64 bytes from 172.217.22.46: icmp_seq=1 ttl=57 time=34.363 ms
64 bytes from 172.217.22.46: icmp_seq=2 ttl=57 time=7.211 ms
64 bytes from 172.217.22.46: icmp_seq=3 ttl=57 time=7.693 ms
```

To implement the SIGINFO functionality in shell scripts, a signal handler routine must be implemented that calls a function which executes when the signal is received. Here is a simple Bourne shell script that echoes "Hello from SIGINFO".

```
#/bin/sh
info() {
    echo "Hello from SIGINFO"
}

trap info SIGINFO
while (true); do
done
```



The output looks like this:

```
bcr@demo:~ % ./siginfo.sh
load: 0.22  cmd: sh 732 [runnable] 2.26r 0.01u 0.75s 8% 2132k
Hello from SIGINFO
```

By default, the shell handles the CTRL+T already by echoing some details about the running process. The information from our own info() function is also displayed. Why not try out SIGINFO on many of the base system utilities that FreeBSD has installed and see if SIGINFO is implemented? A good amount of information is just a keycombination away. Soon, that habit will become a dearly missed functionality when sitting in front of an OS that does not have SIGINFO. ●

BENEDICT REUSCHLING joined the FreeBSD Project in 2009. After receiving his full documentation commit bit in 2010, he actively began mentoring other people to become FreeBSD committers. He is a proctor for the BSD Certification Group and joined the FreeBSD Foundation in 2015, where he is currently serving as vice president. Benedict has a Master of Science degree in Computer Science and is teaching a UNIX for software developers class at the University of Applied Sciences, Darmstadt, Germany.