systemd and the "fd" exhaustion

September 12, 2015 · 4 min

UPDATE 2018/06/24: A fix just landed in *systemd* to fix the related issue.

Few days ago I faced a quite weird error. Well, what was weird was the apparent error message. I wanted to run `journalctl -f` to see the last logs of a server and keep the journal open but it only returned the following error:

```
Failed to get journal fd: Too many open files
```

This can happen, fd exhaustion is a common issue so let's check the current limits and settings to see if there is an actual fd exhaustion; and where it could come from. `ulimit -n` returns 65,535 and a `lsof | wc -l` returns 12,948, we are far away from a fd exhaustion.

The error returned by `journalctl` can't be a fd exhaustion, so what is it? Let's use `strace` to find more verbose errors, as usual.

```
~ # strace -f journalctl -fn
execve("/usr/host/bin/journalctl", ["journalctl", "-fn"], [/* 26 vars */]
brk(0)                  = 0x557694f9c000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory
[...]
openat(AT_FDCWD, "/run/log/journal/c961e28ba6f74af6b63ba50efdb053ba", O_R
getdents(4, /* 3 entries */, 32768) = 88
open("/run/log/journal/c961e28ba6f74af6b63ba50efdb053ba/system.journal",
[...]
inotify_init1(O_NONBLOCK|O_CLOEXEC) = -1 EMFILE (Too many open files)
[...]
writev(2, [{"Failed to get journal fd: Too ma"..., 45}, {"\n", 1}], 2) =
close(5)                = 0
exit_group(1)           = ?
+++ exited with 1 +++
```

Hm interesting, it successfully opened a file but the syscall to `inotify_init1` returned a `EMFILE` error. Let's check another software which uses inotify: `tail -f .`

```
~ # tail -f my.log
lot of stuff
tail: inotify cannot be used, reverting to polling: Too many open files
```

Oh, so it appears that there is too much inotify instances running on the system. The kernel restricts the number of inotify instances with the setting `fs.inotify.max_user_instances` which is set by default to 128.

Let's check the number of running instances per user using the `anon_inode:inotify` symlink in `/proc/*/fd` (*source*):

```
~ # find /proc/*/fd/* -type l -lname 'anon_inode:inotify' -print 2>/dev/n
    128 root
     36 systemd-resolve
     31 messagebus
```

We can confirm that root is, indeed, running out of available inotify instances. The cause of this exhaustion is still a mystery, let's split out the count by command:

```
~ # find /proc/*/fd/* -type l -lname 'anon_inode:inotify' -print 2>/dev/n
      3 root      9841 systemd
      3 root      7207 systemd
      3 root      6428 systemd
      3 root      6021 systemd
      3 root      4275 systemd
      3 root      4001 systemd
      3 root     28636 systemd
      3 root     28591 systemd
...
```

At that time it's important to say that this server runs containers using `systemd-nspawn`, it explains why we see more that one systemd process. Now we note that each systemd process consumes 3 inotify instances; not to mention the inotify instances of other systemd binaries like `systemd-resolved`. This, taking into account, each container will consume at least 4 inotify instances with the user root; as a result, the server will run out of available inotify instances after booting ~32 instances with the default setting.

So let's fix this issue by increasing the upper limit of inotify instances to 256 or more:

```
sysctl fs.inotify.max_user_instances=256
```

Don't forget to persist the change by adding the line into `/etc/sysctl.d/` .

Now `journalctl` should work as expected:

```
~ # journalctl -fn
-- Logs begin at Mon 2015-08-24 12:30:57 UTC. --
Sep 10 09:03:43 host123 systemd[9472]: pam_unix(systemd-user:session): se
...
```

Great!

This task completed I have two comments:
First, ensure that you will not hit inotify instances upper limit when playing with a lot of containers. Then it's sad to still see softwares which return error messages taken out of context.

But as usual, thank you `strace` :)

Enjoy!

systemd