DEBUG: cron keeps piling up in macOS

So, long story... After upgrading to macOS Catalina my years-old automount.sh script running via cron stopped working. It’s been a long enough journey of fixing the script itself (sudo permissions, PATH variable not having some important directories in it when run as a script), but after script was fixed I faced another problem: cron processes keep piling up.

Why is this a problem? Eventually, my Macbook would end up with having more than 10 thousand (!) cron related processes and would just run out of process space — no command can be typed, no app can be started. Only shutdown and power on would fix this.
I’ve been looking at this problem for quite some time, and now that I’m closer to solving it I’d like to share first findings.

**What is this cron thing?**

Don’t remember if I mentioned cron much on [Unix Tutorial](https://www.unixtutorial.org), so here’s a brief summary: cron is a system service that helps you schedule and regularly run commands. It has crontabs: files which list recurrence pattern and the command line to run.

Here’s an example of a **crontab**, each asterisk represents a parameter like “day of the week”, “hour”, “minute”, etc. Asterisk means “every value”, so this below would run my script every minute:

```
* * * * /Users/greys/scripts/try.sh
```

And here’s my automounter script, it runs every 15 minutes (so I’m specifying all the valid times with 15min interval – 0 minutes, 15 minutes, 30 minutes and 45 minutes):

```
0,15,30,45 * * * * /Users/greys/scripts/automount.sh
```

Every user on your Unix-like system can have a crontab (and yes, there’s a way to prohibit cron use for certain users), and usually root or adm user has lots of OS specific tidy-up scripts in Linux and Solaris systems.
The thing with **cron** is it’s supposed to be this scheduler that runs your tasks regularly and then always stays in the shadows. It’s not meant to be piling processes up, as long as your scripts invoked from cron are working correctly.

---

**Debugging cron in macOS**

Turns out, `/usr/sbin/cron` has quite a few options for debugging in macOS:

```
-x debugflag[,...]
```

Enable writing of debugging information to standard output. One or more of the following comma separated debugflag identifiers must be specified:

- `bit` currently not used
- `ext` make the other debug flags more verbose
- `load` be verbose when loading crontab files
- `misc` be verbose about miscellaneous one-off events
- `pars` be verbose about parsing individual crontab lines
- `proc` be verbose about the state of the process, including all of its offspring
- `sch` be verbose when iterating through the scheduling algorithms
- `test` trace through the execution, but do not perform any actions

What I ended up doing is:

**Step 1: Kill all the existing crons**
mcfly:~ greys$ sudo pkill cron

Step 2: Quickly start an interactive debug copy of cron as root

mcfly:~ root# /usr/sbin/cron -x ext,load,misc,pars,proc,sch

When I say “quickly” I’m referring to the fact that cron service is managed by launchd in macOS, meaning you kill it and it respawns pretty much instantly.

So I would get this error:

mcfly:~ root# /usr/sbin/cron -x ext,load,misc,pars,proc,sch  
-sh: kill: (23614) - No such process  
debug flags enabled: ext sch proc pars load misc  
log_it: (CRON 24156) DEATH (cron already running, pid: 24139)  
cron: cron already running, pid: 24139

And the approach I took is kill that last running process and restart cron in the same command line:

mcfly:~ root# kill -9 24281; /usr/sbin/cron -x ext,load,misc,pars,proc,sch  
debug flags enabled: ext sch proc pars load misc  
[24299] cron started  
[24299] load_database()  
    greys:load_user()  
linenum=1  
linenum=2  
linenum=3  
linenum=4
load_env, read <* * * * /Users/greys/scripts/try.sh &> /dev/null>
load_env, parse error, state = 7

I’ll admit: this is probably way too much information, but when you’re debugging an issue there’s no such thing as too much – you’re getting all the clues you can get to try and understand the problem.

In my case, nothing was found: cron would start my cronjob, let it finish, report everything was done correctly and then still somehow leave an extra process behind:

```
[17464] TargetTime=1579264860, sec-to-wait=0
[17464] load_database()
[17464] spool dir mtime unch, no load needed.
[17464] tick(41,12,16,0,5)
user [greys:greys:....] cmd="/Users/greys/scripts/try.sh"
[17464] TargetTime=1579264920, sec-to-wait=60
[17464]    do_command(/Users/greys/scripts/try.sh, (greys,greys,))
[17464] main process returning to work
[17464] TargetTime=1579264920, sec-to-wait=60
[17464] sleeping for 60 seconds
[17473] child_process('/Users/greys/scripts/try.sh')
[17473] child continues, closing pipes
[17473] child reading output from grandchild
[17474] grandchild process Vfork()'ed
log_it: (greys 17474) CMD (/Users/greys/scripts/try.sh)
```
Here’s how the processes would look:

0 17464 17213   0 12:40pm ttys003    0:00.01 /usr/sbin/cron -x ext,load,misc,pars,proc,sch
0 17473 17464   0 12:41pm ttys003    0:00.00 /usr/sbin/cron -x ext,load,misc,pars,proc,sch
0 17476 17473   0 12:41pm ttys003    0:00.00 (cron)
0 17520 17464   0 12:42pm ttys003    0:00.00 /usr/sbin/cron -x ext,load,misc,pars,proc,sch
0 17523 17520   0 12:42pm ttys003    0:00.00 (cron)

How To Avoid Crons Piling Up in Catalina

I’m still going to revisit this with a proper fix, but there’s at least an interim one identified for now: you must forward all the output from each cronjob to /dev/null.

In daily (Linux-based) practice, I don’t redirect cronjobs output because if there’s any output generated – it’s likely an error that I want to know about. cron runs a command, and if there’s any output, it sends an email to the user who scheduled the command. You see the email, inspect and fix the problem.

But in macOS Catalina, it seems this won’t work without further tuning. Perhaps there are some mailer related permissions missing or something like that, but fact is that any output generated by your cronjob will make cron process
keep running (even though your cronjob script has completed successfully).

So the temporary fix for me was to turn my crontab from this:

```
0,15,30,45 * * * * /Users/greys/scripts/automount.sh
* * * * /Users/greys/scripts/try.sh
```

to this:

```
0,15,30,45 * * * * /Users/greys/scripts/automount.sh
> /dev/null 2>&1
* * * /Users/greys/scripts/try.sh > /dev/null 2>&1
```

That’s it for now! I’m super glad I finally solved this — took a few sessions of reviewing/updating my script because frankly I focused on the script and not on the OS itself.

**See Also**

- macOS Catalina
- macOS
- Unix Tutorial Projects: Dell XPS laptop keyboard backlight — has an example of using crontabs