Mac crontab: Creating MacOS startup jobs with crontab, er, launchd

MacOS *crontab* FAQ: How do I run a Unix job (or shell script) through the MacOS *crontab*facility? I keep trying to edit my Mac crontab file, but my Mac won't save my crontab changes, or run my program.

macOS: crontab, launchd, and launchctl

Many years ago (~2012-2014) I found that the Mac `crontab` command was deprecated on MacOS, and the Apple documentation encouraged you to use their *launchd* facility. Here's a blurb from Apple's `crontab` man page:

"Darwin note: Although cron(8) and crontab(5) are officially supported under Darwin, their functionality has been absorbed into launchd(8), which provides a more flexible way of automatically executing commands. See launchctl(1) for more information."

It looked like you could still use the Mac crontab facility, as implied by this note in the MacOS `cron` man page:

"The cron utility is launched by launchd(8) when it sees the existence of */etc/crontab* or files in */usr/lib/cron/tabs*. There should be no need to start it manually."

However, I was never able to get crontab to work under Mac OS X 10.6, so ... in this tutorial I'll go with Apple's suggestion and show you how to run your Unix shell scripts and commands with the MacOS *launchd* facility using the `launchctl` command.

Running a simple command every minute with Mac launchd

For my purposes, I want to run a shell script every minute to ping my websites. If the sites don't respond, I want to be able to notify myself of the problem, perhaps by displaying a dialog from the MacOS Unix shell.

To get this running, I followed the steps shown here.

1) Move to the $HOME/Library/LaunchAgents directory

First, open a Mac Terminal window, then `cd` to this directory:

```
$HOME/Library/LaunchAgents
```

When I dug around in the Apple documentation, I found there are three main directories you can use with `launchd`, and that's how I learned about this directory. Here are your three options:

1. **/Library/LaunchDaemons** - Put your plist scripts in this folder if your job needs to run even when no users are logged in.
2. **/Library/LaunchAgents** - Put your plist scripts in this folder if the job is only useful when users are logged in. (Note: I learned that this has the side-effect of your job being run as 'root' after a system reboot.)
3. **$HOME/Library/LaunchAgents** - Put your plist files in this folder if the job is only useful when users are logged in. (When your plist configuration file is placed here, your job will be run under your username.)

Note that when you use the first two directories shown here, you must use the `sudo`command to edit your files.

To keep this simple and just see how things work initially, my advice is to use the *$HOME/Library/LaunchAgents* folder until you see how things work, then use the other two system folders if/when necessary.

2) Create a Mac plist file to describe your job

Next, create a Mac plist file in this directory to describe the job you want to run. In my case I fired up vi to edit my file:

```
vi com.alvin.crontabtest.plist
```

Following Apple's documentation (and after many errors), I ended up with these contents in my plist file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/P
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.alvin.crontabtest</string>
```

```
  <key>ProgramArguments</key>
  <array>
    <string>/Users/al/bin/crontab-test.sh</string>
  </array>

  <key>Nice</key>
  <integer>1</integer>

  <key>StartInterval</key>
  <integer>60</integer>


  <key>RunAtLoad</key>
  <true/>

  <key>StandardErrorPath</key>
  <string>/tmp/AlTest1.err</string>

  <key>StandardOutPath</key>
  <string>/tmp/AlTest1.out</string>
</dict>
</plist>
```

This plist file can be read as "Run the script */Users/al/bin/crontab-test.sh* every 60 seconds, and redirect the standard output and standard error as shown".

A note about the naming convention: Apple strongly encourages you to use the naming convention I've shown here for (a) your filename and (b) your `label` value. When using the commands that I'm about to show you, you'll refer to the filename and this "label", and they encourage you to follow this naming convention to avoid namespace collisions. Having programmed in Java, this is just like the Java package naming convention, and I have no problems following it.

(You'll see other Mac launchd jobs running when you use the `launchctl list` command below, and I think after that, you'll agree that naming convention is a good idea.)

3) Tell MacOS about your Mac plist launchd file

Next, it's important to know that your MacOS system won't pick up on this change immediately. You have to tell the Mac launchd daemon to *load* it, using the `launchctl`command, like this:

```
launchctl load com.alvin.crontabtest.plist
```

In my case, I just had my */Users/al/bin/crontab-test.sh* shell script write some output to a file in the */tmp* directory so I could debug this process, like this:

```
date >> /tmp/MyLaunchdTest.out
```

After issuing the `launchctl load` command I started getting output from my `date`command. I let it run for several minutes while I checked that everything was working, and then turned it off using this `unload` command:

```
launchctl unload com.alvin.crontabtest.plist
```

4) How Mac launchd works with system reboots

Next, I tested how this works with MacOS system reboots.

In my tests, I found that just the presence of my file in the *$HOME/Library/LaunchAgents*directory was enough to make MacOS load the file and begin running my script every minute. To test this properly, I issued the Mac `launchctl unload` command, like this:

```
launchctl unload com.alvin.crontabtest.plist
```

and then I rebooted my system.

After logging into my Mac after the reboot and checking my output files, I found that my plist script had begun executing every minute.

It's important to note that after a reboot you have to use a slightly different `unload`command than what I showed earlier. In my previous example you didn't need to include the full path to your plist file, but now that the system has started your job for you, you need to unload it by specifically the full path to the file, like this:

```
launchctl unload /Users/Al/Library/LaunchAgents/com.alvin.crontabtest.plist
```

An important note about root and sudo access

It's also **very** important to note that if you placed your Mac plist file in one of the two system directories (*/Library/LaunchDaemons*, */Library/LaunchAgents*), your job will be running as the root user after a system reboot. This means a couple of things:

First, output files created by your script will be owned by the root user.

Second, you'll need to use `sudo` before any of your `launchctl` commands, as shown here:

```
sudo launchctl list | grep 'alvin'
```

If you issue that `launchctl` command like this, without `sudo`:

```
launchctl list | grep 'alvin'
```

you'll never see that your job is running, even though it is. (Because it's owned by root, you're not allowed to see it.)

MacOS launchd, launchctl, and plist resources

I used the following MacOS launchd, launchctl, and plist resources while researching this tutorial:

- Creating launchd daemons and agents
- Scheduling timed jobs with launchd
- And very importantly, if you want to understand the *launchd*vocabulary, the launchd.plist man page is going to be your friend.
- Script management with launchd (not very useful any more)

I also wrote these additional posts after writing this article, when various problems and issues cropped up and I needed more details:

- MacOS launchctl StartInterval OnDemand keys not working
- MacOS launchd plist examples
- MacOS StartInterval and StartCalendarInterval examples

MacOS startup jobs: cron and crontab, launchd and launchctl

In summary, the MacOS launchd facility appears to be a replacement for the standard Unix cron/crontab facility. I believe you can enable crontab to work on MacOS, and I'll show how to do that in a future tutorial. In the meantime, the MacOS launchd/launchctl facility seems to be the future for launching startup jobs, and other jobs we have traditionally run through the cron facility. As such, if you're going to be working a lot on MacOS systems, it behooves you to learn about this new approach.