# 10.3.7 Using anacron

Click one of the buttons to take you to that part of the video.

Use anacron 0:00-0:18

In this demonstration we're going to discuss how anacron works. Most Linux distributions use anacron along with cron to automate the running of tasks. The two services work together to make things happen.

Role and Function of anacron 0:19-1:12

The reason we use anacron with cron is the fact that cron has one key weakness and that is the fact that if the system happens to be down when a particular job was supposed to be run, then that job just gets skipped.

If the system is down consistently when that job was supposed to be run, it never gets accomplished and, as you can imagine, this could be a serious problem for some jobs, such as system backups. If the system happened to be down when those backups were supposed to be run by cron, the backups never happen. That's bad.

Anacron, on the other hand, works around this issue. If a job was scheduled within anacron but then the system is off when that job was supposed to run, then that missed job will be automatically run when the system comes back up.

Just as cron is configured using the crontab files in /etc, the anacron daemon is configured using the /etc/anacrontab file.

anacrontab File 1:13-3:27

The syntax used within the anacrontab file is different than the syntax used within the crontab file. You can see that we have three jobs already defined right down here in the anacrontab file. The first field specifies the recurrence period in days. In other words, how often do we want this job to run? If we have a 1 here, it specifies that the job occurs every day. 7 here means that the job occurs once a week. And down here we have monthly, which specifies that the job recurs once a month.

At first glance, you might be wondering why we don't put 30 down here instead of monthly. If you think about it, some months have 30 days, some months have 31 days, some months--like February--might have 28 or 29 days. If we just used 30 down here, then we would eventually get off in when the job was run. By specifying 'monthly', we account for the fact that months have different numbers of days.

So this first field specifies the recurrence period in days. How often is the job going to run?

The second field right here, and its title is right over here--it doesn't line up very well--the second field right here specifies the delay in minutes. In other words, if the system was down when a particular job here was supposed to run, then the delay in the minutes field specifies how long anacron is going to wait after the system comes back up before it runs that job.

This first job has a delay set for five minutes. If the system was down when this job was supposed to run, then after the system comes back up, anacron is going to wait five minutes--and there's a caveat to that we'll talk about in just a minute-- then it will run that job. This one is 25 minutes and this one is 45 minutes.

If you read through this file, you'll notice that it doesn't actually specify the exact time when the job is going to run. The time when the job can run is configured by this variable right here: START_HOURS_RANGE. Currently, it's set to 3-22, which specifies a time range between 3:00 am and 10:00 pm.

anacron--Time Delay 3:28-5:04

A minute ago when we were talking about the delay period down here, we said that there's a caveat to when these jobs are run. If the system is down when this job is supposed to run, then after the system comes up anacron is going to wait this number of minutes before running this job--kind of.

It's important to note that anacron will also add a random number of minutes to whatever value we specify down here under delay in minutes. The number of minutes that can be added is constrained by this variable right here: RANDOM_DELAY.

Currently, that is set to a value of 45. This causes anacron to add a random number of minutes somewhere between 0 and 45 to the delay minutes specified down here for this skipped job in delay minutes.

Let's suppose this job did not get run because the system was down when it was supposed to run. The system has now come back up. What anacron is going to do is grab a random number between 0 and 45. Say it comes up with 3. It'll take that 3 and add it to the 5 down here that is specified in the delay minutes and wait that many minutes before running this particular job.

It does this to prevent the system from getting choked with backed up jobs when it first comes online. When the system comes online, folks are going to want to start logging in and working, and they would be really irritated if there were 153 jobs queued up waiting to run as they're trying to do that.

By adding a little bit of random delay, it gives the system time to get up and running, folks to get logged in, and then while they're doing their stuff we can start running these missed jobs.

---

## Cooperation Between anacron and cron 5:05-6:17

If you were paying attention, you probably noticed something very interesting about the jobs in this anacrontab file. You'll notice that anacron is working hand in hand with cron to make sure that the daily, weekly, and monthly cron jobs get run.

Within the /etc/cron.d directory, there is a file called 0hourly. We look at it; we see that it causes the cron daemon to run the scripts using the run-parts command right here, at one minute after the hour, every single hour of the day in the cron.hourly sub-directory on /etc.

We do an 'ls' command, however, '/etc/cron.d directory'. There are no crontab files that run the scripts in the cron.daily, cron.weekly, or cron.monthly directory. That is because cron turns over the management of these scripts to anacron. Anacron takes care of the cron scripts in these three directories while the cron daemon takes care of the scripts in the hourly directory.

As you can see, cron and anacron work together to schedule system jobs.

---

## Summary 6:18-6:31

That's it for this demonstration. In this demo we talked about anacron. We first talked about the role and function of anacron. We then looked at the anacrontab file and then we discussed the fact that anacron and cron work together to make sure that jobs get run on the schedule specified.

---

**Copyright © 2022 TestOut Corporation All rights reserved.**