How a cron bug became the de-facto standard

While testing some scheduling edge cases, an old bug was discovered in cron. As long as your crontab schedules' day-of-month and day-of-week fields' values are just simple asterisks (*) or don't contain a * at all, you'll be fine. Otherwise, definitely keep on reading.

Background

If you are an experienced system administrator, you've likely set up your fair share of scheduled jobs using cron and are familiar with the scheduling syntax.

[minute] [hour] [day-of-month] [month] [day-of-week]

And you are well aware of how these five fields are interpreted by cron to determine when your job is supposed to run. That is, minute and hour and month all need to match the current time plus either day-of-month or day-of-week need to match as well. For example 0 12 1 * 1-5 runs at noon on the 1st day of every month and also on every work-weekday. It is basically the union of 0 12 1 * * and 0 12 * * 1-5.

The man page for crontab(5) describes it like this:

```
  The day of a command's execution can be specified in the following two
fields - 'day of month', and 'day of week'. If both fields are restricted
(i.e., do not contain the "*" character), the command will be run when
either field matches the current time. For example, 30 4 1,15 * 5 would
cause a command to be run at 4:30 am on the 1st and 15th of each month, plus
every Friday.
```

Implementation

To better understand the bug, we first have to dig a little into the implementation. The way cron decides whether the time right now matches a crontab schedule can be described with the set notation. If neither day-of-month nor day-of-week are *, cron takes the union (∪) of their values days-of-month ∪ days-of-week. Otherwise cron takes the intersection (∩) of their values days-of-month ∩ days-of-week.

For example, if day-of-month is 10 and day-of-week is *, i.e. one of them is an asterisk, we take the 10th of the month intersected with all days-of-week which results in “the 10th of the month regardless of the day-of-week”. So the schedule fires on exactly one day every month.

If day-of-month is 10 and day-of-week is TUE, i.e. none of them are an asterisk, we take the 10th of the month in a union with all Tuesdays which results in “the 10th of the month plus on all Tuesdays”. So this schedule fires on four, five, or even six days of the month, depending on how many Tuesdays the month has and whether the 10th falls on a Tuesday.

The Bug

Cron's implementation is actually only checking if the very first character of the day-of-month or day-of-week field is an asterisk to decide whether to intersect or to union the day fields. This can cause a problem if either field is a longer token that happens to start with an asterisk. For example, you might think that the following two schedules produce exactly the same result, since the 10 is included in the asterisk wildcard. But they don't.

0 12 *,10 * 2 ≠ 0 12 10,* * 2

The first one's day-of-month starts with an asterisk so cron uses intersect. The schedule fires only on Tuesdays because all-days-of-month ∩ Tuesday = Tuesday. It is the same schedule as 0 12 * * 2.

The second schedule has an asterisk in the day-of-month field, but not as the first character. So cron uses union. The schedule fires every day because all-days-of-month ∪ Tuesday = all-days-of-month. It is therefore the same as 0 12 * * *.

This is admittedly an obscure bug because most people are aware that the asterisk is a wildcard standing for all values. So there is no point in creating a list with a wildcard. However, the problem also shows up when using ranges. You know that 0-59 * * * * is the same schedule as * * * * *. But the following two schedules are not the same.

0 12 1-31 * 2 ≠ 0 12 * * 2

The first one fires every day (same as 0 12 1-31 * * or as 0 12 * * *), and the second schedule fires only on Tuesdays.

This bug is most likely to affect you when using step values. Quick reminder on step values: 0-10/2 means every second value from zero through ten (same as the list 0,2,4,6,8,10), and */3 means every third value. By using an asterisk with a step value for day-of-month or day-of-week we put cron into the intersect mode producing unexpected results.

Most of the time, we choose to use the wildcard to make the cron more legible. However, by now you understand why 0 12 */2 * 0,6 does not run on every uneven day of the month plus on Saturday and Sundays. Instead, due to this bug, it only runs if today is uneven and is also on a weekend. To accomplish the former behaviour, you have to rewrite the schedule as 0 12 1-31/2 * 0,6.

POSIX

POSIX is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. It also describes the behaviour of cron. IEEE 1003.1 specifically says “[if the field] is an asterisk”. It doesn't say “starts”. So this is clearly a deviation from the standard.

Impact

To this day, Vixie cron (now officially called ISC Cron) is the dominant implementation of cron. It only checked the first character for * since its beginning in 1988. It wasn't a big deal back then because

step values weren't supported yet. However, in 1993 step values were introduced with version 3, making it more likley that someone could run into this bug.

The latest versions of Ubuntu, Debian, FreeBSD, OpenBSD, MacOS all include Vixie cron with this behaviour. Red Hat, Fedora, and CentOS have switched from Vixie cron to cronie. However, the schedule-parsing part of cronie is based on Vixie cron with the same problematic implementation.

Vixie cron has been around for almost 30 years and is used everywhere. So if Vixie cron behaves in a certain way, this way is the de-facto standard (even more so than POSIX). I therefore submit that this has become a feature despite the counterintuitive behaviour. Rather than update the code to fix the bug.

Confirmation

Paul Vixie, the creator of Vixie cron, confirmed the bug and agreed that it shouldn't be changed.

“this is certainly a bug, and it comes about because i wrote the parser at the char level rather than the token level, and because the step function syntax is a non-POSIX invention that was added later on.”

and

“i think you're right; fixing this would violate the principle of least astonishment for those rare users who are unknowingly depending on this bug in their current operations.”

Other Implementations

When using other implementations that claim to be cron compatible, be aware that most of them have probably not implemented the first-character-is-asterisk union/intersect behaviour of Vixie cron described in this article.

Simple Test

Here's a simple test to determine if your implementation of cron is affected by this issue. Add the following line to your crontab.

* * *,* * SUN touch /tmp/cron-does-not-have-this-bug

The important part is to not use the current day-of-week. So if today is a Sunday, replace SUN with FRI, for example. Then let it run for a couple of minutes before checking if the file was created. If the file isn't there, your cron does have this problem. Remember to remove the cronjob again when you're done with this test.

Test explanation: If the day-of-week is today, intersected with all days-of-month, it'd result in the cron job running today. If it's not today, the intersection is empty and the job does not run. If it's a fixed version of cron, it would do a union and definitely run today.

From:
https://wiki.plecko.hr/ - **Eureka Moment**

Permanent link:
https://wiki.plecko.hr/doku.php?id=linux:misc:cron:bug

Last update: **2019/10/31 09:14**