Cron – A Beginner's Guide

Table of Contents

1 Colophon

2 Introduction

This is a guide to cron.

It is assumed that the reader is familiar with the Linux command line and at least one text editor. Desktop users will need to open a terminal to execute many of the commands in this guide.

While aimed at Raspberry Pi OS, much will apply to any Linux distribution.

2.1 What's Not Covered

- OS installation and configuration.
- Networking configuration.

2.2 Requirements:

- An internet connection for software update and installation.
- Raspberry Pi (any model).
- For a headless[1] server, VNC or ssh enabled.

2.3 Conventions

```
Text like this indicates input to or output from the command line.
```

`Text like this` also refers to full or partial commands but is not generally intended to be entered into the command line as is.

"SD card" refers equally to full size and micro SD cards. It should also be taken to refer to any boot storage medium in use.

Where "pi" occurs as a username, replace as appropriate.

All non system paths used in this guide follow the Filesystem Hierarchy Standard[2] though this is not mandatory.

Unless stated otherwise "python" refers equally to python 2 and python 3.

1 I.E. no monitor, mouse, or keyboard connected.
2 https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

2.4 References

```
man cron
man crontab
man 5 crontab
```

3 Cron Basics

3.1 What Is Cron?

From the manual page for cron “cron – daemon to execute scheduled commands”.

Cron is a background process started at system boot that can be used to run arbitrary commands at a scheduled time or interval.

On a default installation of Raspberry Pi OS all users have access to its services and, as with any command, a normal unprivileged user should be used unless root privileges are absolutely necessary.

3.2 When To Use Cron

Use cron for the following:

- When a process needs to run at a regular interval.
- When a process needs to run at a set time and/or on a set day.
- When the process does not have any dependencies that it cannot manage itself. Including having access to a running desktop.
- When it does not matter if more than one instance of the process is running at a time unless the process manages this itself.
- When process management tools other than the basic ones provided by the shell[3] are needed.

3.3 When Not To Use Cron

Cron should not be used:

- To start long running processes or services at boot.[4]
- To start process that you wish to manage through systemd or other process management tools.
- To start processes with significant dependencies that they cannot manage themselves.
- Where only one instance of a process can be run at a time unless the process manages this itself.

3 ps, kill, killall, etc.

4 Cron will not restart a failed process until the next scheduled time. There are ways around this but a systemd service is generally a better choice.

3.4 Limitations

Processes started under cron are subject to the following limitations:

- All processes are started in the background.
- Processes are not attached to a terminal – no output will be shown and no input received.
- On a default Raspberry Pi OS installation all output and errors will be discarded unless redirected to a file or an MTA[5] is installed.
- There are significant differences between running a process under cron and running the same process under a logged in shell. See below. A process that runs well under a logged in shell may not run well, or at all, under cron.

3.5 Differences between Cron And A Logged in Shell

Cron does not run the user specific or system wide .bashrc, .bash_aliases, and .profile. Anything set or started in these files will not be available.

Cron does not set any environment variables[6] except the bare minimum:

Variable	**Under cron**	**When Logged In**	**Impact**
$HOME	User’s home directory e.g. `/home/pi`	User’s home directory e.g. `/home/pi`	None
$LOGNAME	Username e.g. `pi`	Username e.g. `pi`	None
$PATH	`/usr/bin:/bin`	`/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games`	Programs and commands not in /usr/bin or /bin and that are not built in to the shell must be called using their full path e.g. /usr/sbin/shutdown
$SHELL	`/bin/sh`	`/bin/bash`	Bash specific features and commands will not be available.
$PWD	User’s home directory e.g. `/home/pi`	Varies depending on `cd` commands issued.	Relative[7] paths and file names may be searched for in the wrong place.
$DISPLAY	Not set	Only set when using the desktop or forwarding X11 over ssh.	Attempts to start desktop applications will fail.

A logged in shell will have many more environment variables set.

5 **M**ail **T**ransfer **A**gent

6 See https://en.wikipedia.org/wiki/Environment_variable

7 Paths and file names not starting with `/`

4 The Crontab File

4.1 Location

Crontabs can be found in `/var/spool/cron/crontabs/` and are named for the user who owns them. They should not be edited directly, instead the `crontab` command should be used.

A user's crontab does not exist until created via the `crontab` command.

Alternative files can exist in the following locations:

- `/etc/crontab`
- `/etc/cron.d`
- `/etc/cron.hourly`
- `/etc/cron.daily`
- `/etc/cron.weekly`
- `/etc/cron.monthly`

For most uses cases it is easier to ignore these as root privileges are required to make changes to them.

4.2 Format

The crontab file consists of two, optional, sections though without the second it will do nothing. Neither sections requires headers or separators.

Comments can be included by starting a line with `#`.

4.2.1 Environment Variables

This section is optional and should come before any crontab entries. It consists of one NAME=value pair per line e.g.

```
SHELL=/bin/bash
```

Environment variables set this way apply to all processes started from the crontab.

4.2.2 Crontab Entries

A crontab entry consists of six fields separated by spaces:

Name	**Purpose**	**Permitted Values**
Minute	Minute to run at	0 to 59 or *
Hour	Hour to run at	0 to 23 or *
Day of Month	Day of month on which to run	1 to 31 or *
Month	Month in which to run	1 to 12 or *
Day of Week	Day of week in which to run	0 to 7 or * Both 0 and 7 map to Sunday
Command	Command to run	Any valid command

The first five fields also accept:

- Ranges: `1-5`
- Lists: `1,2,6`
- Steps: `*/n` e.g. `*/5 or 1-10/3` Steps cannot be used with lists.

The first five fields may be replaced as a group by one of the following. In this case a crontab entry will consist of two fields only.

String	**Equivalent To**	**Notes**
`@reboot`	N/A	Run immediately after system boot. Command is run as soon as the cron daemon is started. Services it depends on (e.g. networking, the desktop, etc.) may not be available
`@hourly`	0 * * * *	
`@midnight`	0 0 * * *	
`@daily`	0 0 * * *	
`@weekly`	0 0 * * 0	
`@monthly`	0 0 1 * *	
`@annually` or `@yearly`	0 0 1 1 *	

4.2.3 Examples

4.2.3.1 Ping google.com every five minutes

```
*/5 * * * * ping -q -c 3 google.com
```

4.2.3.2 Reboot at 03:00 every Sunday

```
0 3 * * 0 /usr/sbin/reboot
```

4.2.3.3 Update a file’s time stamp once a minute

```
* * * * * touch /tmp/foo
```

4.2.3.4 Download a web page on system start

```
@reboot wget http://myserver/myfile
```

5 The `crontab` Command

Or how to edit your crontab.

The `crontab` command allows you to view, edit, and remove your crontab. Usage is a follows:

To edit:	`crontab -e`
To view:	`crontab -l`
To remove:	`crontab -r`

When `crontab -e` is run for the first time it will prompt you to select an editor and create a default crontab for you.

To edit root’s crontab prefix with `sudo` or first login as root.

With root privileges it is possible to edit any user’s crontab by including the `-u username` command option e.g.

```
sudo crontab -u pi -e
```

6 Troubleshooting

6.1 First Find The Error

Redirect at least stderr from the non functioning crontab entry to a file. Normal shell redirection can be used for this e.g.

```
@reboot ping google.com 2>$HOME/ping.log
```

Wait for the next scheduled run and check the log file.

If the log file is empty look for cron related errors in /var/log/syslog:

```
grep cron /var/log/syslog | tail -100 | more
```

6.2 Some Common Issues, Their Causes, And Potential Solutions

This is not an exhaustive list and is in no particular order.

6.2.1 It works in Thonny or in a logged in shell but not cron

Cause	**Fix**
Differences in environment	Find the error then work through the rest of this section. See also 3.5

6.2.2 Command or file not found

Cause	**Fix**
Typo in crontab	Correct the crontab entry
Command is not installed	Install it
Command is not in $PATH	1. Use the full path to the command file e.g. `/usr/sbin/reboot` 2. Set $PATH appropriately in your crontab e.g. `PATH=/usr/sbin:$PATH reboot` or add `PATH=/usr/sbin:$PATH` to the top of your crontab

6.2.3 Output Files Are Not Created Or Updated Or Are Written In The Wrong Place Or A Required File Cannot Be Found

Cause	**Fix**
Using a relative path to the file.	Use the full path to the file e.g. `/home/pi/myfile` instead of just `myfile`
Current/present working directory is not what you or your program expects.	As above
Python is buffering all output and the program has not yet exited	Start python with the `-u` command line argument.

See also 6.2.8

6.2.4 Python Complains That a Method, Class, etc. Is Unknown

Cause	**Fix**
Running under the wrong version of python	Run using the correct version

6.2.5 Python Cannot Import A Module

Cause	**Fix**
Running under the wrong version of python	Run using the correct version
Module installed for the wrong python version	(Re)install for the correct version
Module installed in a virtual environment	Install the module for all users system wide e.g. `sudo pip3 install foo`
Module installed for a single user	As above

6.2.6 Cannot Open Display

Cause	**Fix**
$DISPLAY is not set	Set $DISPLAY appropriately in your crontab e.g. `DISPLAY=:0 lxterminal` or add `DISPLAY=:0` to the top of your crontab
$DISPLAY is set to the wrong value	Fix it
Desktop security settings prevent access	Adjust security setting. Details are outside the scope of this guide.
No desktop[8] is running	Start one. Details are outside the scope of this guide.

8 Actually no X server

6.2.7 Python Cannot Import A Custom Module

Cause	**Fix**
Module is not in the current/present working directory and not in the default locations searched by python.	1. `cd` to the correct directory before starting your script e.g. `cd /home/pi/mydir ; python3 foo.py`[9] 2. Set an appropriate PYTHONPATH environment variable[10] 3. In your python code manipulate sys.path before doing the import.[11] 4. In your python code use os.chdir() before doing the import.[12] 5. Copy or rename your primary script to `__main__.py`, add it and the custom module(s) to a .zip file then pass that zip file to the python interpreter e.g. `python3 /home/pi/mypy.zip`

6.2.8 Permission Denied and Similar Errors

Cause	**Fix**
The user attempting to read or write the file does not have permission to do so.	Adjust permissions on the file. See `man chmod`
The user attempting to read or write the file does not have permission to access the directory containing the file.	Adjust permissions on the directory. See `man chmod`
An attempt to directly run a file that does not have execute permission	Adjust permissions on the file. See `man chmod`
File is on a file system mounted with the `ro` and/or `noexec` mount options	Remount the file system without those options or relocate the file to different file system.

Do not use `chmod` to grant any permissions to files that are normally accessible only by root to non root users.

6.2.9 Cron Doesn't Restart A Failed or Exited Process

This is expected behaviour. Cron is not a service manager like systemd. Processes that exit for any reason will not be restarted until the next scheduled time. If you need restart support consider using a systemd service instead of cron (particularly for `@reboot` jobs) or wrapping the actual command inside a custom shell script that handles the restarts.

9 cd as a separate entry in your crontab will not fix this.
10 See https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH
11 See https://docs.python.org/3/library/sys.html#sys.path
12 See https://docs.python.org/3/library/os.html#os-file-dir

7 Advanced Usage

7.1 Calling a Python, Shell, etc. Script Directly

Most scripts can be called directly from cron by applying the following steps:

1. Ensure the first line of the script is an appropriate shebang[13]

   For python:

   ```
   #!/usr/bin/python3
   ```

   For bash:

   ```
   #!/usr/bin/bash
   ```

   For others refer to their documentation.

2. Use `chmod` to give the file execute permission.

3. Use `/path/to/file` as the command field in your crontab e.g.

   ```
   /home/pi/myscript
   ```

Compiled programs (e.g. C programs) do not require a shebang but do require execute permission.

7.2 Getting Close To A Login Shell

This will not give a true login shell as the process will still run in the background and will not be connected to a terminal.

Simply prefix the desired command with a call to your chosen shell with the `--login` option. For example:

```
/usr/bin/bash --login ping -q -c 3 google.com
```

If the existing command contains special characters and/or I/O redirection it must be enclosed in single quotes or the special characters otherwise escaped:

```
/usr/bin/bash --login 'ping -q -c 3 google.com 2>/tmp/ping.log'
```

Most shells support the --`login` option or the short form equivalent `-l`.

13 https://en.wikipedia.org/wiki/Shebang_(Unix)

7.3 /etc/crontab

In the majority of cases it is preferable to use a user’s crontab via the `crontab` command but it may desirable to use the system wide crontab file in `/etc/crontab`.
`/etc/crontab` differs from a user’s crontab in the following ways:

- It mus be edited with a text editor as it cannot be edited via the crontab command.
- It can only be edited by root or with sudo.
- It contains seven fields rather than six:
    - The first five are the same as in a user’s crontab (see 4)
    - The sixth is the username to run the job under.
    - The seventh is the same as in a user’s crontab (see 4)

7.4 /etc/cron.d

Task specific crontab fragments are stored here. Files must be created or edited as root or with sudo. As with `/etc/crontab`, files are edited with a text editor not the `crontab` command.
Format is the same as `/etc/crontab` (see 7.3).
`/etc/cron.d` is mostly used during package installation when a package requires processes to be run by cron.

7.5 /etc/cron.hourly, /etc/cron.daily, etc.

Scripts for cron to run at the specified time (see 4.2.2). These are not crontab files.

- Files must have execute permission.
- Files must have an appropriate shebang.
- Files will be run as root. It is the responsibility of the script to change to a different user if required.
- Output will be sent to cron’s default. It is the responsibility of the script to send it elsewhere if required.

7.6 Increasing Logging

In the default configuration cron only logs the start of a job. Additional logging is available and can be enabled as follows:

1. Backup `/etc/default/cron`
2. Open `/etc/default/cron` in your preferred text editor. This will require root privileges.
3. Locate the line the reads
   ```
   #EXTRA_OPTS=””
   ```
4. Remove the leading `#`
5. Insert `-L N` between “” when `N` is one of 0, 1, 2, 4, 8 or the sum of some or all of those numbers. See below for their meanings.
6. Save and close
7. Restart cron
   ```
   sudo systemctl restart cron
   ```

Log Level	**Meaning**
0	Disable logging
1	Log start of jobs
2	Log end of jobs
4	Log jobs that fail
8	Log the process ID of child processes

The default is a log level of 1. To enable more than one level add them together.

8 Change Log

2023-02-16

Initial Release