SCHEDULING TASKS USING CRON

Edmond Lascaris and Anne Jessup - Creative Commons – 15 June 2021

OVERVIEW

In this lesson we are going to learn how to automate routine tasks on the Raspberry Pi. This is helpful is ever you need to back up files, upload content to a web server or sample data from a sensor.

Cron is a program that was designed to schedule routine tasks (commands, scripts or programs) on Linux systems, such as the Raspberry Pi. The command **crontab** is used to edit the list of scheduled tasks. Each user on the computer has their own **cron** table, so you can think of **cron** as your own personal assistant.

In this lesson we will automate the running of the atm_sensor_get.py python program using **cron**.

LEARNING OBJECTIVES

- Running python programs from the **Terminal**
- Learn how to find the directory **path** to applications and programs on your computer
- Learn how to use the text editing software **nano**
- Learn how to create scheduled tasks in **cron**

RUNNING PYTHON PROGRAMS FROM THE TERMINAL

In this example we are going to run the atm_sensor_get.py program using the **Terminal**. Up to now, we have only run python programs from within the **Python3 IDLE3** environment. To run our atm_sensor_get.py program we will use the command **python3**.

1. **Open the Terminal and navigate to your project's directory**

- From the Raspberry Pi top bar menu click on the **Terminal** icon.
- Once in your default home directory (/home/pi) enter **ls** to list all contents.
- Find your project's directory (e.g. botanica-park-lake) and use the change directory (**cd**) command to enter the directory (e.g. **cd botanica-park-lake**)
- Enter the command **ls** to list all directory contents.

```
pi@raspberrypi: ~/botanica-park-lake
File  Edit  Tabs  Help
pi@raspberrypi:~ $ ls
atm_sensor_get.py   Desktop    melb_weather_1.py  Public         Videos
bom                 Documents  Music              techschool
bom.txt             Downloads  Pictures           Templates
botanica-park-lake  MagPi      projects           test_grep.txt
pi@raspberrypi:~ $ cd botanica-park-lake/
pi@raspberrypi:~/botanica-park-lake $ ls
atm.html            data.txt      __pycache__            test1.txt
atm_sensor_get.py   index.html    rainbow-lorikeet.jpg   testPython.py
birds.html          mystyles.css  README.md              timestamp.py
pi@raspberrypi:~/botanica-park-lake $
```

- You should see the python file **atm_sensor_get.py**
- To run this file, enter the command **python3 atm_sensor_get.py**
    - **python3** – name of application
    - **atm_sensor_get.py** – script or program to run
- The **printouts** we can see in the Terminal are the outputs from **print**() statements within our python code.

```
pi@raspberrypi:~/botanica-park-lake $ python3 atm_sensor_get.py
Date_stamp,        Temp,Pres,    Bat
2021-Jun-17 05:57,6.6,100.053,4.066


<h1>Atmospheric Conditions</h1>
<p>The current temperature is 6.6</p>
<p>The current atmospheric pressure is 100.053</p>
<p>The current battery voltage is 4.066</p>

pi@raspberrypi:~/botanica-park-lake $
```

FINDING THE DIRECTORY PATH TO APPLICATIONS AND PYTHON PROGRAMS

In this example we will find the path of the python3 application and scratch on the Raspberry Pi. A **path** shows the location where a software application can be found within the directory structure on your computer. To find the **path** for an application we use the Terminal command **which** followed by the name of the software application.

2. **Open the Terminal and navigate to your project's directory**

- From the Raspberry Pi top bar menu click on the **Terminal** icon.
- Once in your default home directory (/home/pi) enter **ls** to list all contents.
- Find your project's directory (e.g. botanica-park-lake) and use the change directory (**cd**) command to enter the directory (e.g. **cd botanica-park-lake**)
- Enter the command **ls** to list all directory contents.

```
pi@raspberrypi: ~/botanica-park-lake
File  Edit  Tabs  Help
pi@raspberrypi:~ $ ls
atm_sensor_get.py   Desktop    melb_weather_1.py  Public        Videos
bom                 Documents  Music              techschool
bom.txt             Downloads  Pictures           Templates
botanica-park-lake  MagPi      projects           test_grep.txt
pi@raspberrypi:~ $ cd botanica-park-lake/
pi@raspberrypi:~/botanica-park-lake $ ls
atm.html            data.txt      __pycache__            test1.txt
atm_sensor_get.py   index.html    rainbow-lorikeet.jpg   testPython.py
birds.html          mystyles.css  README.md              timestamp.py
pi@raspberrypi:~/botanica-park-lake $
```

- To find path to a current directory enter the command **pwd** (present working directory).
- This is helpful when we need to know the path to a program we have written.
- In this case, the **path** to the botanical-park-lake directory is **/home/pi/botanica-park-lake**
- The full path to the atm_sensor_get.py program is **/home/pi/botanica-park-lake/atm_sensor_get.py**

```
pi@raspberrypi:~/botanica-park-lake $ pwd
/home/pi/botanica-park-lake
pi@raspberrypi:~/botanica-park-lake $
```

3. **Finding the path to applications and Linux commands.**

- To find the **path** to a software application we use the Terminal command **which**

- To find the path to the **python3** (binary executable file) we enter the command **which python3**
- The output in the Terminal is **/usr/bin/python3**
- The **path** to the python3 application is **/usr/bin**

```
pi@raspberrypi:~/botanica-park-lake $ which python3
/usr/bin/python3
pi@raspberrypi:~/botanica-park-lake $
```

- We can find the path to other commonly used Linux commands.
- For example, the command **ls** is also an application. To find its path enter **which ls**
- The directory path to the **ls application** is found in the **/bin** directory (short for **bin**ary files)

```
pi@raspberrypi:~/botanica-park-lake $ which ls
/bin/ls
```

- Try this with other commands such as **cat**, **cd**, **touch** and **pwd**. Do you see a pattern?
- You can find other commands here: Linux commands - Raspberry Pi Documentation
- You can also find the path to other programming applications, such as **Scratch**
- Enter the command **which scratch**

```
pi@raspberrypi:~/botanica-park-lake $ which scratch
/usr/bin/scratch
```

4. Finding an application or file within a directory

- Sometimes when we use the Terminal command which to get the path for an application, we also want to navigate to the directory to see the application within the directory.
- To navigate to the **/usr/bin** directory enter the command **cd /usr/bin**

```
pi@raspberrypi:~/botanica-park-lake $ cd /usr/bin
pi@raspberrypi:/usr/bin $
```

- To list all the files and directories enter the command **ls**
- Unfortunately, in this directory there are more than 1000 files (1335 files to be exact).
- You can count the number of files with the command **ls -1 | wc**

- **ls -1** - this command lists all files and directories in one column (number one -1, not the letter L)
- **|** - this is the **pipe symbol** (above the Enter key – is a vertical line). It takes the output from one command and makes it the input for another command.
- **wc** - this command is short for Word Count

```
pi@raspberrypi:/usr/bin $ ls -1 | wc
   1335    1335   13532
pi@raspberrypi:/usr/bin $
```

5. Narrowing down file search results using the Terminal command grep

- We can narrow our search down a little using another command called **grep**
- In the same directory enter the command **ls -1 | grep python3**
- This command will highlight only those entries that have the file name python3
  - **ls -1** - list all files as a single column (-1 option)
  - **|** - pipe symbol
  - **grep python3** - search for file names containing the text "python3"

```
pi@raspberrypi:/usr/bin $ ls -1 | grep python3
arm-linux-gnueabihf-python3.7-config
arm-linux-gnueabihf-python3.7m-config
arm-linux-gnueabihf-python3-config
arm-linux-gnueabihf-python3m-config
dh_python3
idle-python3.7
python3
python3.7
python3.7-config
python3.7m
python3.7m-config
python3-config
python3m
python3m-config
pi@raspberrypi:/usr/bin $
```

- Using the **| pipe** symbol we can also do a word count at the very end to see how many python3 files there were with the command **ls -1 | grep python3 | wc**
- As you can see, the | pipe symbol is very handy.

```
pi@raspberrypi:/usr/bin $ ls -1 | grep python3 | wc
     14      14     276
pi@raspberrypi:/usr/bin $
```

USING THE TERMINAL TEXT EDITOR NANO

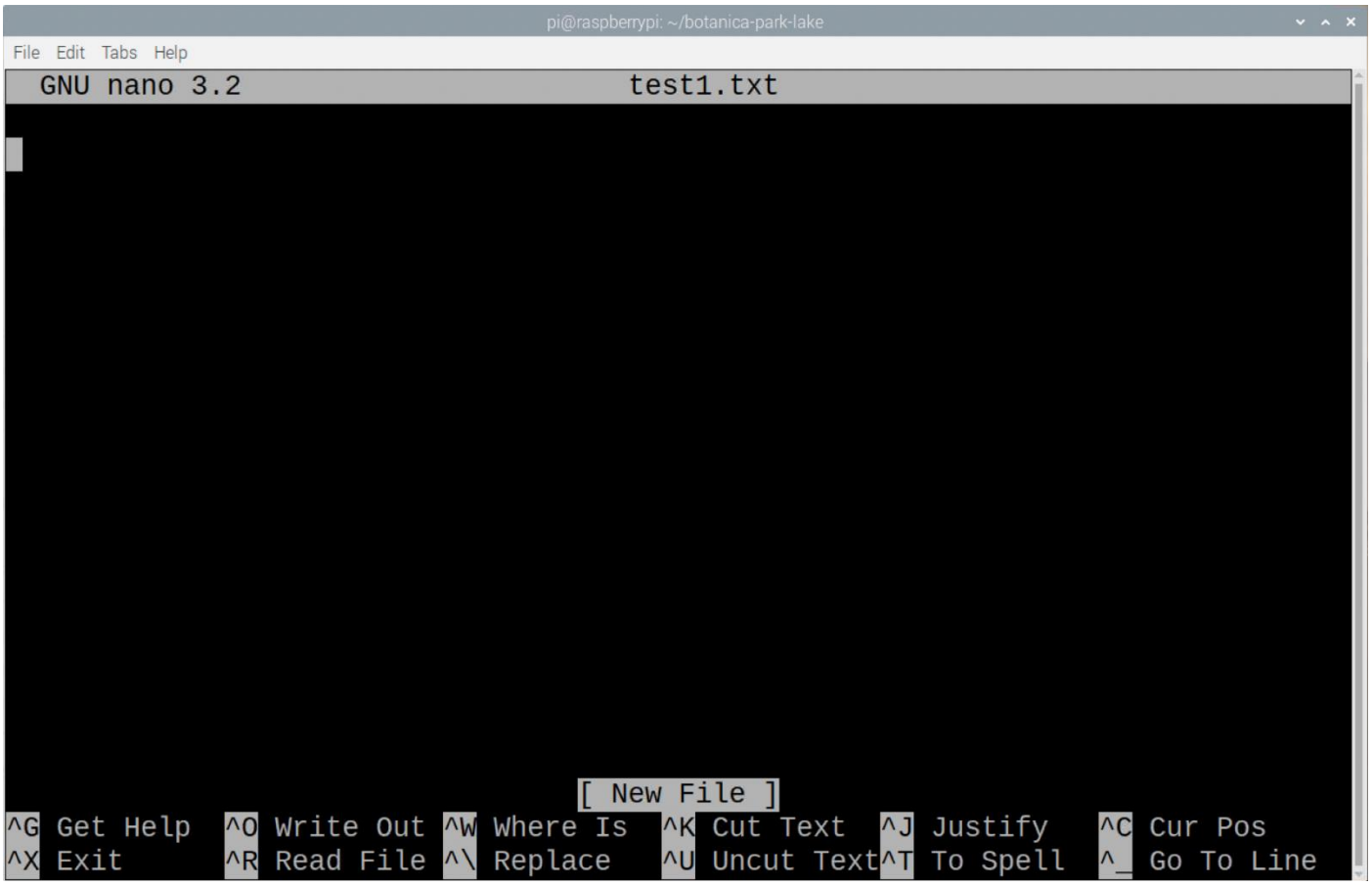
Nano is a text file editor that is used for creating and modifying text-based files. It is a simple editor and is widely used on Linux platforms. We need to learn how to use nano so that we can write scheduling tasks. In this example we will use nano to create a new file, add some text to the file and then save the file.

6. Create a new file using nano

- Navigate to your project directory using the Terminal command **cd**
- To create an empty file named **test1.txt** that can be edited in nano enter the command **nano test1.txt**

```
pi@raspberrypi:~/botanica-park-lake $ nano test1.txt
```

- This will open the **nano test editor** to an empty document.



- Enter text using the keyboard. You won't be able to use the mouse.
- To navigate simply use the keyboard **arrow keys**.

```
  GNU nano 3.2                    test1.txt                    Modified

The nano text editor is used to edit text based files on the Raspberry Pi.

To launch nano enter the command nano followed by the file name.

Additional functions in nano can be activated using the CTRL key.

To see more functions enter CTRL+G

To get back to the main menu enter CTRL+X

^G Get Help   ^O Write Out ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File ^\ Replace    ^U Uncut Text^T To Spell   ^_ Go To Line
```

7. Saving files in nano

- When you have finished editing your document enter **CTRL+X** keys to Exit.
- You will be prompted to **Save** your file.
- Enter **y** for yes and press Enter. Lowercase y is fine.

```
Save modified buffer?  (Answering "No" will DISCARD changes.)
 Y Yes
 N No           ^C Cancel
```

- You will be asked if the file is to be saved as test1.txt
- Accept the name test1.txt and press **Enter**.

```
File Name to Write: test1.txt
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T To Files
```

- When the file is saved you will be returned to the main Terminal window.

LEARN HOW TO CREATE SCHEDULED TASKS IN CRON

Cron is a tool to configure scheduled tasks in Linux. In this example we will use the command **crontab -e** to create one scheduled task. We will configure **cron** to run our python program **atm_sensor_get.py** once every hour so that we can start to collect atmospheric data.

8. Running cron for the first time

- To run Cron enter the Terminal command **crontab -e**
- The first time you run crontab you will need to **select an editor**.
- Choose the relevant number to select **nano** and press Enter.
- Your selection options may different to those shown below.

```
pi@raspberrypi:/var/www/html/aquarium $ crontab -e
no crontab for pi - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/vim.tiny

Choose 1-3 [2]:
```

- When the **Cron table** opens you should see the following comments (**#**) in the first few lines of the file.
- Comments will not be executed. If required, they can be deleted to make the cron table file less cluttered.

```
  GNU nano 3.2                /tmp/crontab.GVjlNb/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
                              [ Read 25 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

9. Setting when to run cron task

- The makeup for a **cron** entry is made up of 6 components:
    1. minute (0-59)
    2. hour (0-23)
    3. day of month (1-31)
    4. month of year (1-12)
    5. day of week (0-6 – Sunday to Saturday)
    6. command to be executed
- If a number is replaced with an *** (asterix),** then that rule applies for every unit
- Here are some examples related to the set time for a task to run:
    - **10 2 * * *** – 2:10am every day

- **00 15 2 * *** - 3pm on the 2nd of every month
- **05 10 * * *** - 10:05am every day
- **03 05 * * *** - 5:03am every day
- **00 15 * * 5** – Every Thursday at 3:00pm
- **05 * * * *** - 5 minutes past the hour, every hour

- We will run our program at **5 minutes past the hour**, every hour so the time rule is **05 * * * ***

10. Entering the command for a cron task

- We would like to run our atm_sensor_get.py file using Python3.
- The full command would therefore be **python3 /home/pi/botanica-park-lake/atm_sensor_get.py**
- The full path needs to be given for our python program atm_sensor_get.py
- We could also give the full path to the python3 application.
- In this case the full command line would be **/usr/bin/python3 /home/pi/botanica-park-lake/atm_sensor_get.py**

11. Creating a complete cron entry

- Now that we have all the details we need for our cron entry we can enter it in one line
- Scroll using the arrow keys to the last entry in the Cron table
- Enter the following line: **5 * * * * python3 /home/pi/botanica-park-lake/atm_sensor_get.py**
- This will then run the python program at 5 minutes past the hour, every hour.

```
# m h  dom mon dow   command
5 * * * * python3 /home/pi/botanica-park-lake/atm_sensor_get.py

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell   ^_ Go To Line
```

12. Saving and viewing cron table entries

- To **save** your cron table, use the same method as for the nano application.

- Save with **CTRL+X**. Then reply **y** for Yes. And then press **Enter** to accept the default file name.

```
File Name to Write: /tmp/crontab.GVjlNb/crontab
^G Get Help        M-D DOS Format     M-A Append         M-B Backup File
^C Cancel          M-M Mac Format     M-P Prepend        ^T To Files
```

- To view all your currently saved scheduled tasks in cron enter the Terminal command **crontab -l**

```
pi@raspberrypi:~ $ crontab -l
```

13. Checking to see if cron is working

- When the atm_sensor_get.py program runs it automatically updates the **data.txt** file.

- You can see the last time this file was updated by either entering the **ls -l** command. This would show the date and time the file was last updated.

```
pi@raspberrypi:~/botanica-park-lake $ ls -l
total 88
-rw-r--r-- 1 pi pi   167 Jun 17 12:22 atm.html
-rw-r--r-- 1 pi pi   888 Jun 17 05:57 atm_sensor_get.py
-rw-r--r-- 1 pi pi   414 Apr  1 03:58 birds.html
-rw-r--r-- 1 pi pi  2607 Jun 17 12:22 data.txt
```

- Alternatively, you can enter **cat data.txt** and look at the last time the file was modified.

- If there are many entries in the data.txt file, you can print out the most recent entries at the end of the file with the command **tail data.txt**

```
pi@raspberrypi:~/botanica-park-lake $ tail data.txt
2021-Jun-17 04:22,7.16,99.991,4.064
2021-Jun-17 05:22,7.14,100.012,4.069
2021-Jun-17 05:57,6.6,100.053,4.066
2021-Jun-17 06:22,6.6,100.053,4.066
2021-Jun-17 07:22,7.25,100.091,4.062
2021-Jun-17 08:22,8.29,100.12,4.059
2021-Jun-17 09:22,9.58,100.161,4.065
2021-Jun-17 10:22,0,0,4.065
2021-Jun-17 11:22,14.91,100.19,4.062
2021-Jun-17 12:22,20.34,100.13,4.067
pi@raspberrypi:~/botanica-park-lake $
```

- Congratulations on completing this lesson. In our next lesson we will start to graph our data using plotly.