

Crontab Documentation

What is a crontab application?

A crontab app is a scheduler which the user enters the exact date and time a command shall be executed, along with the command which shall be run. Each command is stored in an external file which holds the details of every crontab.

A single crontab holds the following details (in this order):

- Minute - in the range 0-59.
- Hour - in the range 0-23.
- Day of the month - in the range 1-31.
- Month - in the range 1-12.
- Day of the week - day of the week stored as a number; week starts on Sunday which is 0 and ends on Saturday which is 6 (though Sunday can also be 7)
- Command - finally, this is the command the user wants executed on the date/time it follows.

Crontab job example:

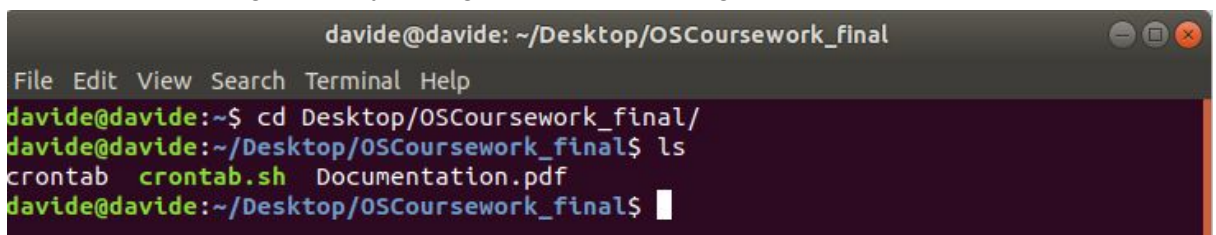
```
30 * * 4 6 /foo/command.sh
```

The shell program 'command.sh' will be run every hour at half past on Saturdays (day 6), but only during April.

Run the program

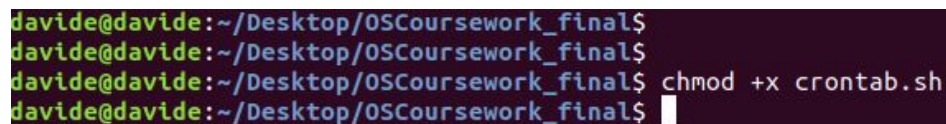
First of all, we have to be sure that the our file (the crontab.sh file) is an executable. To make the file executable, we have to:

1. Move in the working directory through the terminal using the cd command

A terminal window titled 'davide@davide: ~/Desktop/OSCoursework_final' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the user navigating to the Desktop/OSCoursework_final directory and listing its contents. The files listed are crontab, crontab.sh, and Documentation.pdf.

```
davide@davide: ~/Desktop/OSCoursework_final
File Edit View Search Terminal Help
davide@davide:~$ cd Desktop/OSCoursework_final/
davide@davide:~/Desktop/OSCoursework_final$ ls
crontab  crontab.sh  Documentation.pdf
davide@davide:~/Desktop/OSCoursework_final$
```

2. Make the file executable using the chmod +x command

A terminal window showing the user making the crontab.sh file executable using the chmod +x command.

```
davide@davide:~/Desktop/OSCoursework_final$
davide@davide:~/Desktop/OSCoursework_final$
davide@davide:~/Desktop/OSCoursework_final$ chmod +x crontab.sh
davide@davide:~/Desktop/OSCoursework_final$
```

3. Execute the script: ./ExecutableFileName.format

```
davide@davide:~/Desktop/OSCOURSEWORK_final$  
davide@davide:~/Desktop/OSCOURSEWORK_final$  
davide@davide:~/Desktop/OSCOURSEWORK_final$ ./crontab.sh
```

4. Enjoy our script

```
davide@davide:~/Desktop/OSCOURSEWORK_final$ ./crontab.sh  
Welcome to the Mycrontab application  
Indicate the number linked to a specific command  
  
List of crontab commands:  
1 - Display jobs  
2 - Insert a job  
3 - Edit a job  
4 - Remove a job  
5 - Delete all jobs  
6 - Study the documentation for the contrab command  
7 - Exit  
  
Command:
```

How the script works

The application will run until the user decides to exit it. It asks the user to choose between one of the options provided in the menu to interact with the application.

The menu will provide 7 different options:

- 1: Displays the crontab jobs
- 2: Allows the user to insert a job
- 3: Allows the user to edit a job
- 4: Allows the user to remove a job
- 5: Deletes all jobs.
- 6: Study the documentation for the crontab commands. // This is an extra option the user can use. It explains the functionalities of a crontab app to the user.
- 7: Exit - Closes the application

To display the jobs, the “sed” command has been used. It lists the jobs in the crontab file and gives them a number.

To add a command the user will be prompted to insert each value individually and validate it by checking if the number is within the range the number entered should be. If the user leaves the input empty, an error will be displayed and the user will be asked to enter a valid input. As for the command to carry out, we can only check if the user has entered it, not whether syntax is correct or if it's a file which exists.

If the user wants to have a job that will be run any at any time of the given option, a ‘-1’ will be asked to be inserted. That will be then printed as ‘*’ in the crontab file. Once the user has inserted all the fields required, the application will return to the main menu.

A function called “Validate” is found at the beginning of the script, which is used to check that the user is inserting a valid value when adding/editing a job. That function checks:

1. The first argument is not empty when the function is called, which is the user’s input
2. If user’s input is ‘-1’ to treat it as a ‘any time’ (*) option
3. Delimiters for a valid entry, so the function checks if it is within the range of these 2 values. E.g. For minute the field must be greater than -1 and less than 60.

If the crontab file does not exist when the user tries to insert a job, the file is created in the same directory as the shell program. If the user tries to display the jobs when there is no file, an error is sent from the terminal saying that there is no file to be displayed.

To edit a job, the user will be prompted with a list of the jobs with a number relative to it’s line in the crontab file. The user should insert the number of the job they want to edit. A menu will be displayed where the user can choose which field to edit, enter the value they want in in the selected field then insert the new value.

Editing of the crontab file is achieved with the “sed” command, which will first delete the whole line, redirect the rest of the file into a temporary file and then move/append the new command in the crontab file, at which point the temporary file is deleted. The full command is shown as follows:

```
"sed "${NOLINE}d" crontab > file.temp && mv file.temp crontab".
```

To remove a single job, the same mechanism of editing a job has been used. The user will be shown every job via the terminal and asked to insert the line number of the job they want to delete. The jobs will be put in a temporary file which the “sed” command then finds the line number the user entered and removes it from the temporary file. The contents of this temporary file then overwrite the crontab file.

To remove all the jobs, the command “eval > crontab “ has been used which simply empties the crontab file.

To exit the application, the user must enter ‘7’ on the main menu.

How we worked to solve the problem

The coursework has been approached with a teamwork attitude. We first tried to understand the crontab mechanism and how to recreate its functionalities. Davide created and developed a base of the structure and the main functionalities to be implemented at the beginning. As the project developed, more issues were occurred we tried to split the tasks. Simone initially took care of the display and remove tasks, while Taylor and Davide worked to add a task. According to our method, the function to remove a task became part of editing a task, thanks to Simone, and the file handling to add jobs was made possible by Davide. Though at this point there was no input validation, which was completed for each individual input by Taylor.

The final task we faced was editing a job. We combined the purposes of the other tasks to make this as efficient as possible by: listing the jobs, asking the user which job to edit, receiving an input from the user, validating it and then erasing the old job from the file and outputting the new command with the edited value to the crontab file.

Keywords of the source code

In the source, we used different commands for the manipulation of the crontab file and interaction with the user. To understand the structure of the code, find the main commands used in our scripts below:

- **Sed:** used for text substitution, deletion, search and insertion of any data inside a file.
- **Eval:** it concatenates its arguments into a single string, joining the arguments with spaces, then executes that string as a bash command. Eval executes the command in the current shell environment rather than creating a child shell.
- **Echo:** used to display line of text/string that is passed as an argument.
- **Read:** bash command that reads an input.

Date: 08/11/19

Version: 1.0