User-friendly task scheduler for crontab

**Project motivation**

As a team we aim to simplify the use of command-line commands that aim to schedule tasks for the user, by building a user-friendly interface for a 'task scheduler', which otherwise is not very user-friendly.

**Description**

This project aims to make task-scheduling in Unix OS systems much simpler and user-friendly. For this project we are using '**cron**' “*a Unix, solaris, Linux utility that allows tasks to be automatically run in the background at regular intervals by the cron daemon.”*

For the management of the task & their respective scheduling, we are using '**crontab**' (in short for: "CRON TABLE"), which is a file that contains the schedule of cron entries to be run and at specified times by the user.

Cron job (or cron schedule) is a specific set of execution instructions specifying day, time and command to execute. crontab can have multiple execution statements within itself.

**How it works**

1. Run the script by changing the current directory to the one where the file is located and then by typing the following command into the console/terminal: *./crontab-ui.sh*

2. After submitting the command you are presented with a menu with numbered options to manage your tasks.

3. Select an option by typing in the console the corresponding number to that command, example: “*1. Display all jobs*” will display all current active scheduled jobs once you enter “1” to the console.

**Available commands**

|  |  |
| --- | --- |
| **Command** | **Description** |
| 1 | Display crontab jobs |
| 2 | Insert a job |
| 3 | Edit a job |
| 4 | Remove a job |
| 5 | Remove all jobs |
| 9 | Exit |

**Usage Examples**

First and foremost, you must run the script by typing the following in to the console/terminal: *./crontab-ui.sh*

You are presented with a menu with numbered options to manage your tasks.

Select one of the following options by typing in the console the corresponding number to that command from the table “Available commands” shown above.

* **Display crontab jobs [1]:**

If you have typed “1” then you will obtain an output (list-like) of all of the current active crontab jobs. Or receive an error message if there are no jobs.

**Example**:

Command: echo ’hello’.

Running: on every 59 minute/s, every 23 hour/s, on every 31 day of month, on every 12 month, every 0 day of the week

…

* **Inserting a job [2]:**

If you have typed “2” then you will be presented with a series of questions to configure your to-be scheduled task: whether you want a custom schedule or use a preset schedule command, when you want your task to run & what will the task do.

Using Custom Schedule Commands: (Command 1)

- Enter the minutes ( 0 – 59 ) or \* for any

- Enter hour ( 0 – 23 ) or \* for any

- Enter the day ( 1 – 31 ) or \* for any

- Enter the day of the month ( 1 - 12 ) or \* for any

- Enter the weekday ( 0 Mon – 6 Sun) | \* for any

- Enter the name of the command

...

Using Pre-set Schedule Commands: (Commands 2)

- Please select from the list of pre-set commands below:

- @reboot - Run once, at startup

- @yearly or @annualy - Run once a year

- @monthly - Run once a month

- @weekly - Run once a week

- @daily or @midnight - Run once a day

- @hourly - Run once an hour

- Enter pre-set command to use (ie: @reboot):

- Enter command to install:

…

If you have passed all of the error handling we have implemented your task will be successfully inserted.

* **Editing a job [3]:**

If you have typed “3”, you will be presented with list of all of the jobs and a assigned number to each job . You will be prompted select a command number. You will be then asked to change the scheduled task to whatever you want through a similar set of “config” questions like in command above “*Insert a job [2].”* If you have passed all of the error handling we have implemented your task will be successfully edited.

**Example**:

Command No 1: echo ’hello’. Running: on every 59 minute/s, every 23 hour/s, on every 31 day of month, on every 12 month, every 0 day of the week

Command No 2: echo ’goodbye’. Running: on every 58 minute/s, every 22 hour/s, on every 30 day of month, on every 12 month, every 0 day of the week

Select command to be edited:

…

*command [2]*

* **Remove a job [4]:**

If you have typed “4”, similar to *Editing a job [3]* you will be presented with list of all of the commands and a number assigned to each job. You will be prompted select a command number to be removed. The exisitng command number will then be removed

**Example:**

Command No 1: echo ’hello’. Running: on every 59 minute/s, every 23 hour/s, on every 31 day of month, on every 12 month, every 0 day of the week

Command No 2: echo ’goodbye’. Running: on every 58 minute/s, every 22 hour/s, on every 30 day of month, on every 12 month, every 0 day of the week

Select command to be removed:

...

* **Remove all job [5]:**

If you have typed ”5”, all of the scheduled tasks will be removed.

* **Exit task scheduler [9]:**

If you have typed “9”, you will exit the task scheduler user interface.

**Created By**

* Miguel Bacharov
* Valeri Vladimirov
* Mihail Yonchev

**How we split the tasks between each other**

Miguel Bacharov did the following tasks:

* Created the functionality for Edit a job
* Helped with the Documentation file
* Helped with the Error Handling in the project
* Made improvements to the Edit a job and Remove a job functions. Such as listing all the commands first with a given number and removing/editing by the given number.

Mihail Yonchev did the following tasks:

* Created the functionality for Remove all jobs
* Created the functionality for Insert a job
* Improved the Error Handling of the project
* Helped with the design of the project

Valeri Vladimirov did the following tasks:

* Created the functionality for Display all jobs
* Created the functionality for Remove job
* Designed the menu
* Worked on the Documentation File

**Team Working**

Even though we split the tasks between each other, everyone worked on all of the parts. When someone needed help on something, we worked it out together as a team. We communicated everyday on discord, a messenger group chat by giving ideas on how we can further improve the design and functionality of the project. We had meetups every Monday in the Operating Systems practical and sometimes in the library to work additionally on the project. We also created a private git repository for the project where we made regular commits.