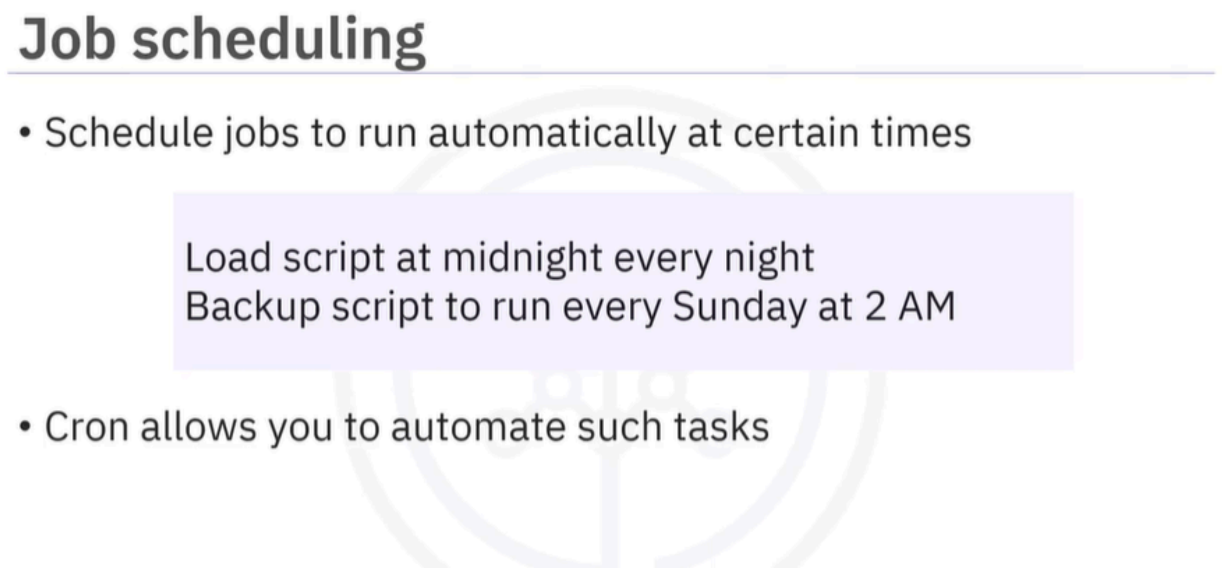
Scheduling Jobs using Cron

Sure! Let's talk about scheduling jobs using Cron in simple terms.

Cron is like a helpful assistant that can automatically run tasks for you at specific times. Imagine you have a friend who reminds you to water your plants every day at 8 AM and to take out the trash every Sunday at 10 AM. In the same way, Cron can be set up to run commands or scripts on your computer at scheduled times, like running a backup every night or loading data every day at midnight.

To set this up, you use something called a crontab, which is like a calendar where you write down what tasks you want to run and when. Each task has a specific format that tells Cron when to do it. For example, if you want to append the current date to a file every Sunday at 3:30 PM, you would write it in a certain way in the crontab. Once you save it, Cron takes care of the rest!



Job scheduling

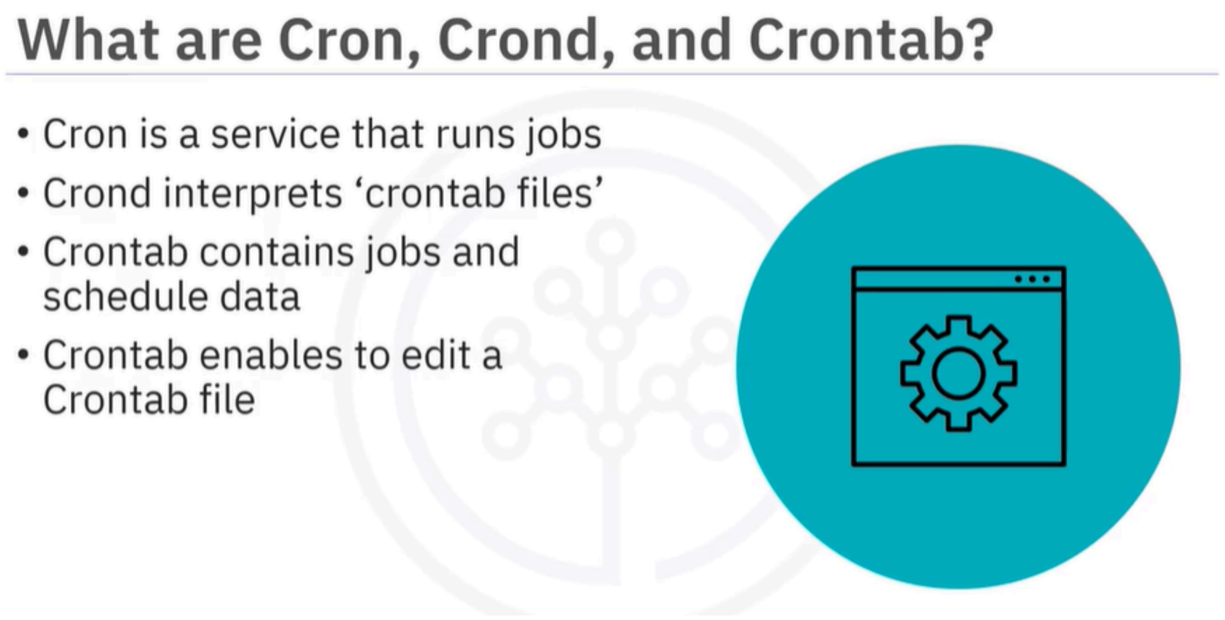
- Schedule jobs to run automatically at certain times

Load script at midnight every night
Backup script to run every Sunday at 2 AM

- Cron allows you to automate such tasks

- Whether you are a system administrator or a data engineer or even a developer, there may be times when you want to schedule certain jobs to run automatically at certain times. For example, you may want to schedule a load script to run every day at midnight, and a backup script to run every Sunday at

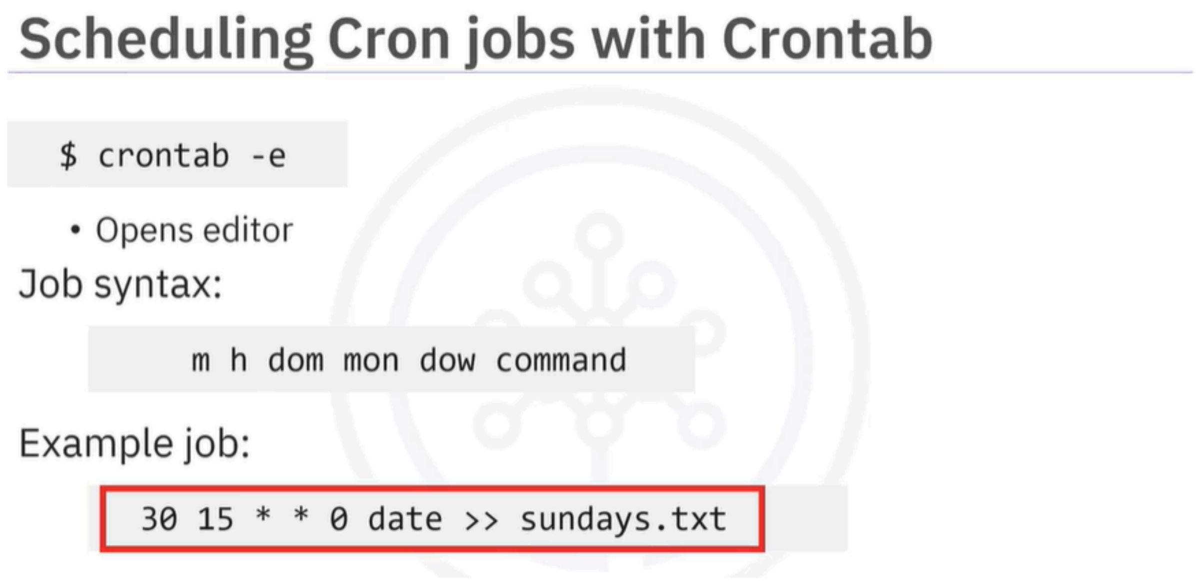
2 AM. The cron utility on Linux and Unix-like operating systems allows you to do just that.



What are Cron, Crond, and Crontab?

- Cron is a service that runs jobs
- Crond interprets ‘crontab files’
- Crontab contains jobs and schedule data
- Crontab enables to edit a Crontab file

- The cron utility on Linux and Unix-like operating systems allows you to do just that. Cron is the general name of the tool that runs scheduled jobs consisting of shell commands or shell scripts. Crond is the daemon or service that interprets “crontab files” every minute and submits the corresponding jobs to cron at scheduled times. A crontab, short for “cron table,” is a file containing jobs and schedule data. Crontab is also a command that invokes a text editor to allow you to edit a crontab file.



Scheduling Cron jobs with Crontab

```
$ crontab -e
```

- Opens editor

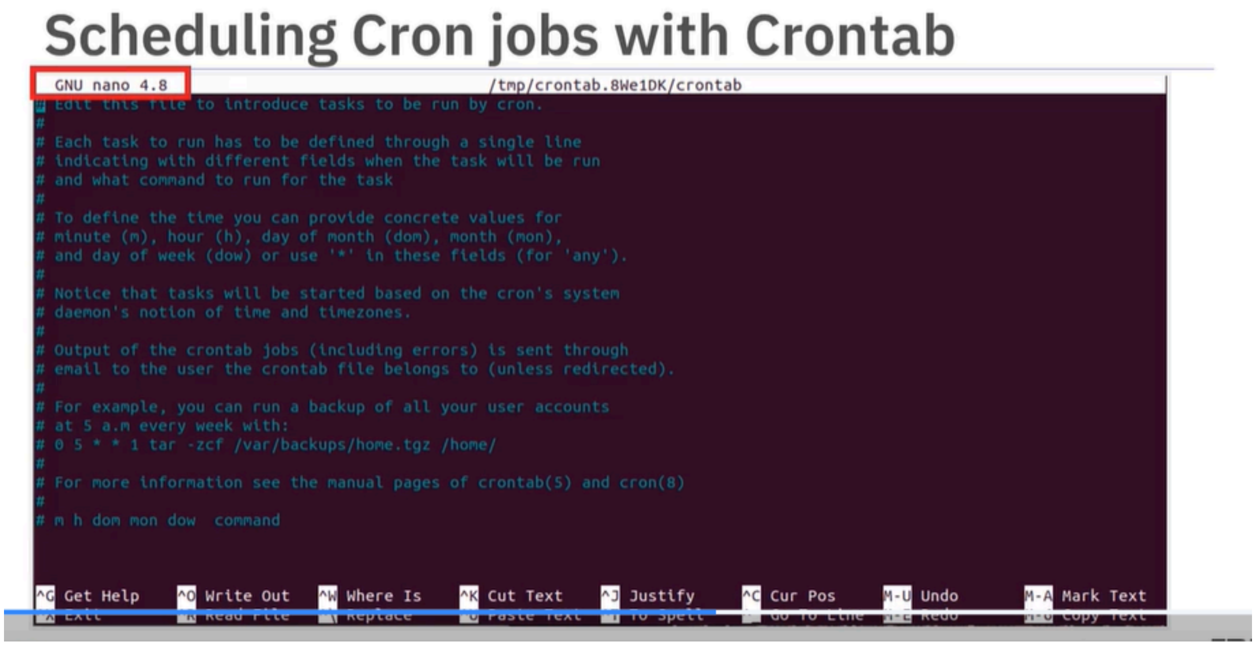
Job syntax:

```
m h dom mon dow command
```

Example job:

```
30 15 * * 0 date >> sundays.txt
```

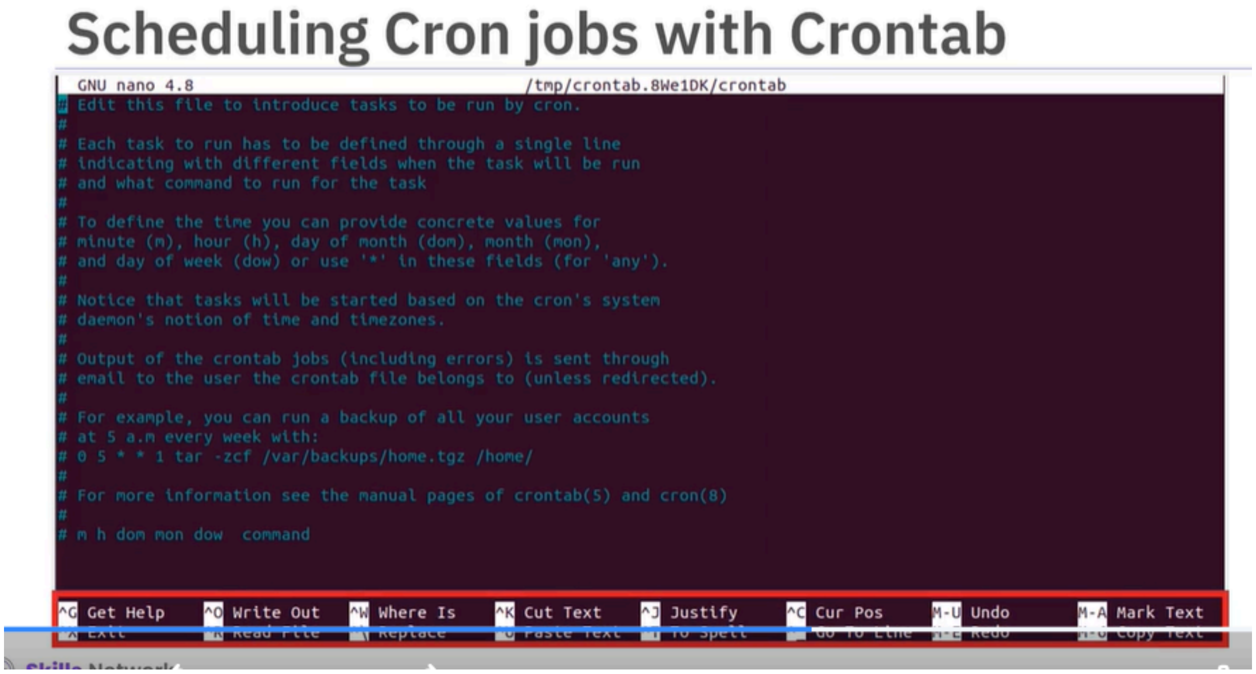
- Entering crontab “minus e” on the command line opens the default text editor. Using the editor, you can specify a new schedule and a command, which has the following syntax: “command” can be any shell command, including a call to a shell script. The symbols stand for minute, hour, day of month, month, and day of week. All five positions must have either a numeric entry or an asterisk, which is a wildcard symbol that means “any.” For example, the following syntax means: append the current date to the file ‘sundays.txt’ at 15:30 every Sunday. Closing the editor and saving the changes adds the job to the cron table.



Scheduling Cron jobs with Crontab

```
  GNU nano 4.8                          /tmp/crontab.8We1DK/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/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text
```

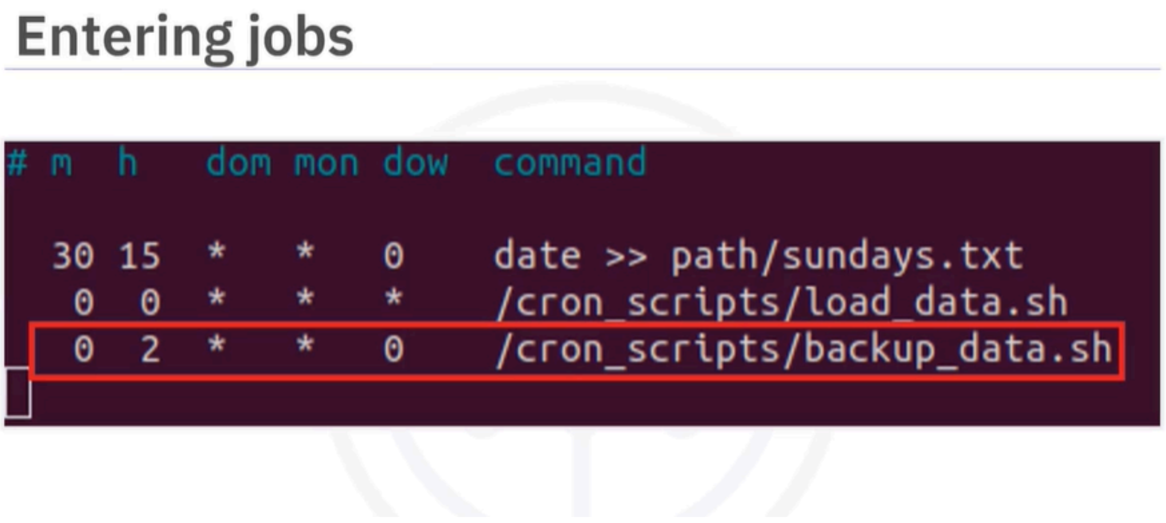


Scheduling Cron jobs with Crontab

```
  GNU nano 4.8                          /tmp/crontab.8We1DK/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/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text
```

- Let's take a closer look at each of these steps. Entering crontab “minus e” on the command line opens the default text editor. In this case, the default editor is GNU nano. Conveniently, instructions for setting up cron jobs are included here as comments. Instructions for using the editor are also included, but you likely will only need “control x” in this context.

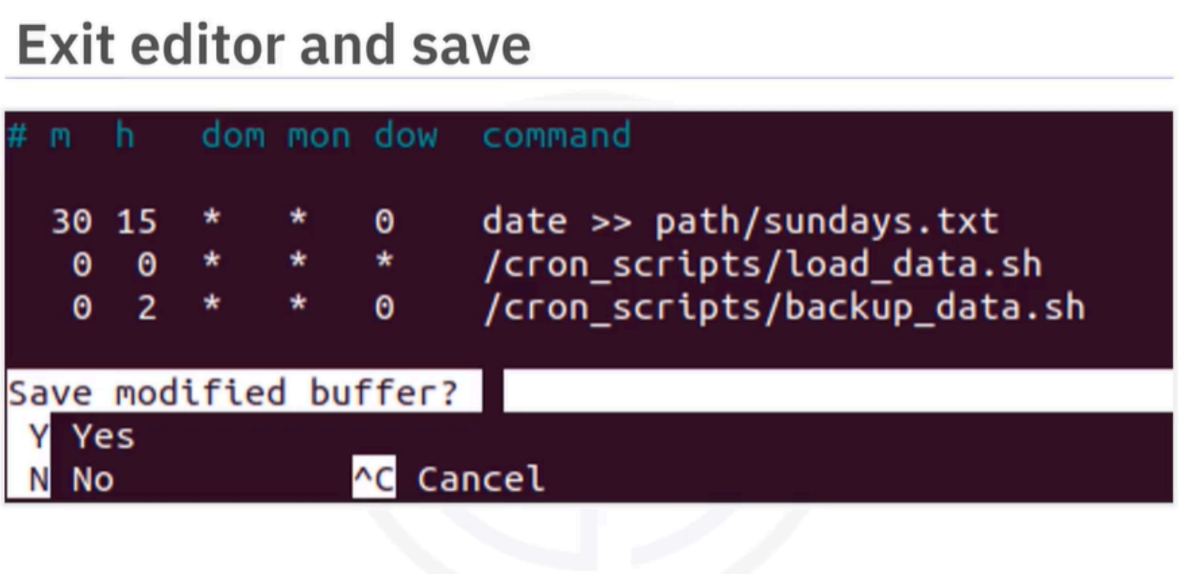


Entering jobs

```
# m  h   dom mon dow  command

  30 15  *   *   0    date >> path/sundays.txt
   0  0  *   *   *    /cron_scripts/load_data.sh
   0  2  *   *   0    /cron_scripts/backup_data.sh
```

- Here I have entered three example cron jobs. Notice that extra spaces are ignored, so to improve the readability you can align your entries as columns under the header. The first entry instructs cron to append the current date to the file 'sundays.txt,' at 15:30 every Sunday. The next line specifies a "load data" shell script to run at midnight every day, while The last line results in cron running the "backup" data shell script to run at 2 AM on Sundays. To save the job, first type "control x" to exit the editor, and then enter "y" to save your changes.



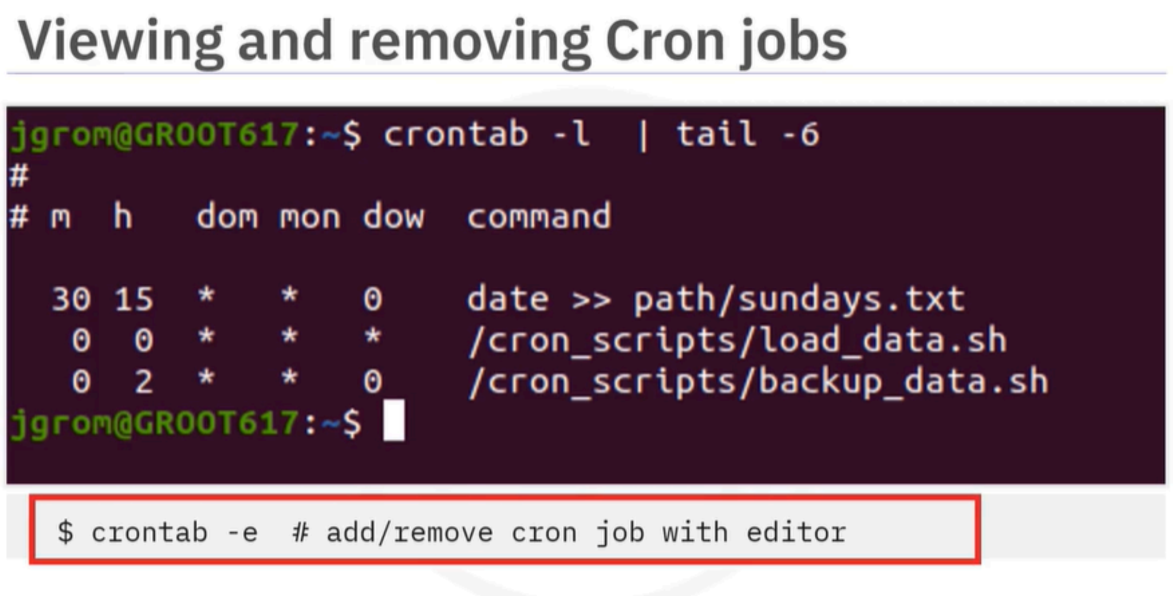
Exit editor and save

```
# m   h    dom mon dow   command

  30 15   *   *   0     date >> path/sundays.txt
   0  0   *   *   *     /cron_scripts/load_data.sh
   0  2   *   *   0     /cron_scripts/backup_data.sh

Save modified buffer?
 Y Yes
 N No               ^C Cancel
```

- The jobs are now in production! Running crontab with the “l” option returns a list of all cron jobs and their schedules.



Viewing and removing Cron jobs

```
jgrom@GROOT617:~$ crontab -l  | tail -6
#
# m  h   dom mon dow  command

  30 15  *   *   0    date >> path/sundays.txt
   0  0  *   *   *    /cron_scripts/load_data.sh
   0  2  *   *   0    /cron_scripts/backup_data.sh
jgrom@GROOT617:~$
```

```
$ crontab -e  # add/remove cron job with editor
```

- I used “tail” to avoid returning all of the comments from the crontab file. To remove a job, simply invoke the crontab editor, delete the corresponding line in the crontab file, and save the changes.

How can you apply cron jobs in your daily tasks?

can apply cron jobs to automate various daily tasks on your computer. Here are a couple of practical examples:

1. **Backup Files**: You can set up a cron job to automatically back up important files every night at a specific time. For instance, you could schedule a script that copies your documents to an external drive or cloud storage at 2 AM every day.

   Example crontab entry:

   ```
   0 2 * * * /path/to/backup_script.sh
   ```

2. **Data Processing**: If you have a script that processes data (like cleaning or transforming data files), you can schedule it to run every hour. This way, you ensure that your data is always up-to-date without having to remember to run the script manually.

   Example crontab entry:

   ```
   0 * * * * /path/to/data_processing_script.sh
   ```

By using cron jobs, you can save time and ensure that important tasks are completed regularly without manual intervention. Would you like to know more about how to set up a specific cron job?You can apply cron jobs to automate various daily tasks on your computer. Here are a couple of practical examples:

1. **Backup Files**: You can set up a cron job to automatically back up important files every night at a specific time. For instance, you could schedule a script that copies your documents to an external drive or cloud storage at 2 AM every day.

   Example crontab entry:

   ```
   0 2 * * * /path/to/backup_script.sh
   ```

2. **Data Processing**: If you have a script that processes data (like cleaning or transforming data files), you can schedule it to run every hour. This way, you ensure that your data is always up-to-date without having to remember to run the script manually.

   Example crontab entry:

   ```
   0 * * * * /path/to/data_processing_script.sh
   ```

By using cron jobs, you can save time and ensure that important tasks are completed regularly without manual intervention.