At Command in Linux

Posted May 4, 2020 • 6 min read

`at` is a command-line utility that allows you to schedule commands to be executed at a particular time. Jobs created with `at` are executed only once.

In this article, we will explain how to use `at` and its companion utilities `batch`, `atq`, `atrm` to view, delete, and create jobs to be executed at a later time.

Installing `at`

Depending on the distribution, `at` may or may not be present on your Linux system.

If `at` is not installed, you can easily install it using the package manager of your distribution.

- **Install `at` on Ubuntu and Debian**

  ```
  $ sudo apt update
  $ sudo apt install at
  ```

- **Install `at` on CentOS and Fedora**

  ```
  $ sudo yum install at
  ```

Once the program is installed make sure `atd`, the scheduling daemon is running and set to start on boot:

```
$ sudo systemctl enable --now atd
```

How to Use the `at` Command

The simplified syntax for the `at` command is as follows:

```
at [OPTION...] runtime
```

The `at` command takes the date and time (`runtime`) when you want to execute the job as a command-line parameter, and the command to be executed from the standard input.

Let's create a job that will be executed at 9:00 am:

```
$ at 09:00
```

Once you hit `Enter`, you'll be presented with the `at` command prompt that most often starts with `at>`. You also see a warning that tells you the shell in which the command will run:

```
Output
warning: commands will be executed using /bin/sh
at>
```

Enter one or more command you want to execute:

```
at> tar -xf /home/linuxize/file.tar.gz
```

When you're done entering the commands, press `Ctrl-D` to exit the prompt and save the job:

```
Output
at> <EOT>
job 4 at Tue May  5 09:00:00 2020
```

The command will display the job number and the execution time and date.

There are also other ways to pass the command you want to run, besides entering the command in the `at` prompt. One way is to use echo and pipe the command to `at`:

```
$ echo "command_to_be_run" | at 09:00
```

Another option is to use Here document :

```
$ at 09:00 <<END
 command_to_be_run
 END
```

To read the commands from a file instead of the standard input, invoke the command with `-f` option following by the path to the file. For example, to create a job that will run the script `/home/linuxize/script.sh`:

```
$ at 09:00 -f /home/linuxize/script.sh
```

By default if the command produces output, `at` will send an email including the output to the user once the job is completed. Invoke `at` with the `-M` option to suppress the email notification:

```
$ at 09:00 -M
```

Use the `-m` to send an email even if there is no output:

```
$ at 09:00 -m
```

`batch` Command

`batch` or its alias `at -b` schedules jobs and executes them in a batch queue when the system load level permit. By default, the jobs are executed when the system load average is below 1.5. The value of the load can be specified when invoking the `atd` daemon. If the system load average is higher the specified one, the jobs will wait in the queue.

To create a job with `batch`, pass the commands you want to execute:

```
$ echo "command_to_be_run" | batch
```

Specifying the Execution Time

The `at` utility accepts a wide range of time specifications. You can specify time, date, and increment from the current time:

- **Time** - To specify a time, use the `HH:MM` or `HHMM` form. To indicate a 12-hour time format, use `am` or `pm` after the time. You can also use strings like `now`, `midnight`, `noon`, or `teatime` (16:00). If the specified time is passed, the job will be executed the next day.

- **Date** - The command allows you to schedule job execution on a given date. The date can be specified using the month name followed by the day and an optional year. You can use strings, such as `today`, `tomorrow`, or weekday. The date can be also indicated using the `MMDD[CC]YY`, `MM/DD/[CC]YY`, `DD.MM.[CC]YY` or `[CC]YY-MM-DD` formats.

- **Increment** - `at` also accepts increments in the `now + count time-unit` format, where `count` is a number and `time-unit` can be one of the following strings: `minutes`, `hours`, `days`, or `weeks`.

Time, date and increment can be combined, here are few examples:

- Schedule a job for the coming Sunday at a time ten minutes later than the current time:

  ```
  $ at sunday +10 minutes
  ```

- Schedule a job to run at 1pm two days from now:

  ```
  $ at 1pm + 2 days
  ```

- Schedule a job to run at 12:30 Oct 21 2020:

  ```
  $ at 12:30 102120
  ```

- Schedule a job to run one hour from now:

  ```
  $ at now +1 hours
  ```

You can also specify a time and date in the `[[CC]YY]MMDDhhmm[.ss]` using the `-t` option. Here is an example:

```
$ at -t 202005111321.32
```

Specifying Queue

By default, the jobs created with `at` are scheduled in the queue named `a` and jobs created with `batch` are scheduled in the `b` queue.

Queries can have a name from `a` to `z` and `A` to `Z`. Queues with lower letters run with lower niceness, which means they have priority over those with higher letters.

You can specify the queue with the `-q` option. For example, to set a job in the `L` queue, you would run:

```
$ at monday +2 hours -q L
```

Listing Pending Jobs

To list the user's pending jobs run the `atq` or `at -l` command:

```
$ atq
```

The output will list all jobs, one per line. Each line includes the job number, date, time, queue letter, and username.

```
Output
9     Tue May  5 12:22:00 2020 a linuxize
12  Wed Oct 21 12:30:00 2020 a linuxize
15  Tue May  5 09:00:00 2020 a linuxize
6     Tue May  5 09:00:00 2020 a linuxize
13  Mon May  4 23:08:00 2020 a linuxize
11  Wed Jul  1 10:00:00 2020 a linuxize
4     Tue May  5 09:00:00 2020 a linuxize
```

When `atq` is invoked as an administrative user, it will list the pending jobs of all users.

Removing Pending Jobs

To remove a pending job invoke the `atrm` or `at -r` command followed by the job number. For example, to remove the job with number nine, you would run:

```
$ atrm 9
```

Restricting Users

The `/etc/at.deny` and `/etc/at.allow` files allow you to control which users can create jobs with `at` or `batch` command. The files consist of a list of usernames, one user name per line.

By default, only the `/etc/at.deny` file exists and is empty, which means that all users can use the `at` command. If you want to deny permission to a specific user, add the username to this file.

If the `/etc/at.allow` file exists only the users who are listed in this file can use the `at` command.

If neither of the files exists, only the users with administrative privileges can use the `at` command.

Conclusion

The `at` utility reads commands from standard input and executes them at a later time. Unlike crontab , jobs created with `at` are executed only once.

For more information about all available options of the `at` command type `man at` in your terminal.

If you have any questions, feel free to leave a comment.