Open in app ↗

Sign In

Patrick Blom   Follow

Jan 9, 2020 · 2 min read · ▶ Listen

🔖 Save   𝕏   f   in   🔗

Linux: What does '1>/dev/null 2>&1' actually mean

If you're quite new to the world of Linux many things will be new and become confusing, because Linux offers you so many variations of control which can be combined in several ways.

One thing of these is the control of the output from a command which runs in the terminal. It quite often happens that you want to prevent any output of a command e.g. in a cronjob or if run a command in background. This makes a lot of sense because if you're running a command in background you simply don't want that it throws all information and error to the terminal your're actually working with.

At this point file descriptors come into place. Every Unix process has to support at least the three common descriptors defined by the POSIX application programming interface.

- **0** *Standard Input*

- **1** *Standard Output*

- **2** *Standard Error*

In combination with the possibility to ⌐̶̶̶̶̶̶̶ with the ">" we can simply explain what "1>/dev/null 2>&1" actually mea⌐̶̶̶̶̶̶̶ command from the terminal e.g. G-

👏 65   💬 1

edit this could look like this·

patrick@Linux:

| 2>&1

Lets explain every part of the command.

- **/usr/bin/gedit** is the program we want to start

- **1** is the file descriptor for *Standard Output*

- **>** is for forwarding

- **/dev/null** is a path to a black hole where any data sent, will be discarded (This could also be the path to a file)

- **2** is the file descriptor for *Standard Error*

- **>** is for forwarding

- **&** is the symbol for file descriptor (without it, the following 1 would be considered as a filename)

- **1** is the file descriptor for *Standard Output*

So in a sentence "1>/dev/null 2>&1" after a command means, that every *Standard Error* will be forwarded to the *Standard Output* and this will be also forwarded to a black hole where all information is lost.

Swift   Linux   Linux Tutorial   Shell   Beginners Guide