# How can I redirect and append both stdout and stderr to a file with Bash?

1362

343

To redirect *stdout* to a truncated file in Bash, I know to use:

```
cmd > file.txt
```

To redirect *stdout* in Bash, appending to a file, I know to use:

```
cmd >> file.txt
```

To redirect both *stdout* and *stderr* to a truncated file, I know to use:

```
cmd &> file.txt
```

How do I redirect both *stdout* and *stderr* appending to a file? `cmd &>> file.txt` did not work for me.

linux bash redirect stream pipe

edited Dec 17 '15 at 16:27
Jahid
**14.3k** 4 63 86

asked May 18 '09 at 4:19
flybywire
**100k** 173 369 484

32 I would like to note that &>outfile is a Bash (and others) specific code and not portable. The way to go portable (similar to the appending answers) always was and still is >outfile 2>&1 – TheBonsai May 18 '09 at 4:48

## 7 Answers

1. `>>file.txt` : Open `file.txt` in append mode and redirect `stdout` there.
2. `2>&1` : Redirect `stderr` to *"where `stdout` is currently going"*. In this case, that is a file opened in append mode. In other words, the `&1` reuses the file descriptor which `stdout` currently uses.

edited Mar 9 '17 at 14:55

Fritz
**661** 6 21

answered May 18 '09 at 4:23

Alex Martelli
**646k** 132 1053 1288

29 works great! but is there a way to make sense of this or should I treat this like an atomic bash construct? – flybywire May 18 '09 at 8:15

174 It's simple redirection, redirection statements are evaluated, as always, from left to right. >>file : Red. STDOUT to file (append mode) (short for 1>>file) 2>&1 : Red. STDERR to "where stdout goes" Note that the interpretion "redirect STDERR to STDOUT" is wrong. – TheBonsai May 18 '09 at 8:55

28 It says "append output (stdout, file descriptor 1) onto file.txt and send stderr (file descriptor 2) to the same place as fd1". – Dennis Williamson May 18 '09 at 9:07

2 @TheBonsai however what if I need to redirect STDERR to another file but appending? is this possible? – arod Jun 2 '13 at 22:26

38 if you do `cmd >>file1 2>>file2` it should achieve what you want. – Woodrow Douglass Sep 6 '13 at 21:24

331

There are two ways to do this, depending on your Bash version.

The classic and portable (**Bash pre-4**) way is:

```
cmd >> outfile 2>&1
```

A nonportable way, starting with **Bash 4** is

```
cmd &>> outfile
```

- decide if portability even to Bash pre-4 is a concern (then use classic way)
- no matter which syntax you use, not change it within the same script (confusion!)

If your script already starts with `#!/bin/sh` (no matter if intended or not), then the Bash 4 solution, and in general any Bash-specific code, is not the way to go.

Also remember that Bash 4 `&>>` is just shorter syntax — it does not introduce any new functionality or anything like that.

The syntax is (beside other redirection syntax) described here: http://bash-hackers.org/wiki/doku.php/syntax/redirection#appending_redirected_output_and_error_output

edited Mar 23 '14 at 11:24 — Mathias Bynens — **109k** 39 179 229

answered May 18 '09 at 4:42 — TheBonsai — **10.4k** 3 17 14

6 I prefer &>> as it's consistent with &> and >>. It's also easier to read 'append output and errors to this file' than 'send errors to output, append output to this file'. Note while Linux generally has a current version of bash, OS X, at the time of writing, still requires bash 4 to manually installed via homebrew etc. – mikemaccana May 20 '13 at 9:30

I like it more because it is shorter and only tweoi places per line, so what would for example zsh make out of "&>>"? – Phillipp Feb 17 '16 at 14:20

Also important to note, that in a cron job, you have to use the pre-4 syntax, even if your system has Bash 4. – hyperknot May 18 '17 at 10:03

3 @zsero cron doesn't use bash at all... it uses `sh` . You can change the default shell by prepending `SHELL=/bin/bash` to the `crontab -e` file. – Ray Foss Jun 5 '18 at 20:45

82

In Bash you can also explicitly specify your redirects to different files:

```
cmd >log.out 2>log_error.out
```

Appending would be:

```
cmd >>log.out 2>>log_error.out
```

6 Redirecting two streams to the same file using your first option will cause the first one to write "on top" of the second, overwriting some or all of the contents. Use ***cmd >> log.out 2> log.out*** instead. – Orestis P. Dec 11 '15 at 14:33

3 Thanks for catching that; you're right, one will clobber the other. However, your command doesn't work either. I think the only way to write to the same file is as has been given before `cmd >log.out 2>&1` . I'm editing my answer to remove the first example. – Aaron R. Dec 11 '15 at 15:36

61

In Bash 4 (as well as ZSH 4.3.11):

```
cmd &>outfile
```

just out of box

edited May 20 '13 at 8:47

mikemaccana
**45.7k** 48 241 311

answered Mar 27 '12 at 18:24

A B
**1,348** 1 14 22

1 @AoeAoe: This actually works in Bash 4 too. – mk12 Sep 6 '12 at 21:11

2 @all: this is a good answer, since it works with bash and is brief, so I've edited to make sure it mentions bash explicitly. – mikemaccana May 20 '13 at 8:47

10 @mikemaccana: TheBonsai's answer shows bash 4 solution since 2009 – jfs Mar 27 '14 at 17:56

42

This should work fine:

```
your_command 2>&1 | tee -a file.txt
```

It will store all logs in *file.txt* as well as dump them on terminal.

edited Mar 3 '16 at 18:35

answered Dec 12 '15 at 6:17

Try this

23

```
You_command 1>output.log  2>&1
```

Your usage of &>x.file does work in bash4. sorry for that : (

## Here comes some additional tips.

0, 1, 2...9 are file descriptors in bash.

0 stands for `stdin` , 1 stands for `stdout` , 2 stands for `stderror` . 3~9 is spare for any other temporary usage.

Any file descriptor can be redirected to other file descriptor or file by using operator `>` or `>>` (append).

Usage: **<file_descriptor> > <filename | &file_descriptor>**

Please reference to http://www.tldp.org/LDP/abs/html/io-redirection.html

edited Mar 10 '18 at 3:47
antzshrek
**3,213** 3 18 35

answered Apr 10 '14 at 5:56
Quintus.Zhou
**608** 5 13

Your example will do something different than the OP asked for: It will redirect the stderr of `You_command` to stdout and the stdout of `You_command` to the file `output.log` . Additionally it will not append to the file but it will overwrite it. – pabouk May 31 '14 at 12:38

Correct: ***File descriptor*** could be any values which is more than 3 for all other files. – Itachi Dec 25 '14 at 6:46

5 Your answer shows the most common output redirection error: redirecting STDERR to where STDOUT is currently pointing and only after that redirecting STDOUT to file. This will not cause STDERR to be redirected to the same file. Order of the redirections matters. – Jan Wikholm Jan 4 '15 at 12:51

1 does it mean, i should firstly redirect STDERROR to STDOUT, then redirect STDOUT to a file. `1 > output.log 2>&1` – Quintus.Zhou Mar 4 '15 at 6:10

1 @Quintus.Zhou Yup. Your version redirects err to out, and at the same time out to file. – Alex Yaroshevich Mar 8 '15 at 23:22

```
(cmd 2>&1) >> file.txt
```

This spawns a subshell, so it's less efficient than the traditional approach of `cmd >> file.txt 2>&1`, but this approach feels more natural and understandable to me:

1. Redirect stderr to stdout.
2. Redirect the new stdout by appending to a file.

Also, the parentheses remove any ambiguity of order, especially if you want to pipe stdout and stderr to another command instead.

edited Feb 15 at 16:16

answered Feb 15 at 16:11

jamesdlin
**28.7k** 6 64 100

Ok. 4 years later you get the recognition. That is how it is with art, you have time to die before getting any recognition ;o) – stefgosselin Mar 2 at 17:44

**protected** by gniourf_gniourf Dec 24 '17 at 8:26

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?