

Get current directory or folder name (without the full path)

Asked 15 years, 3 months ago Modified 2 months ago

Viewed 999k times



1175

How could I retrieve the current working directory/folder name in a bash script, or even better, just a terminal command.



`pwd` gives the full path of the current working directory, e.g. `/opt/local/bin` but I only want `bin`.



bash

shell

Share

Improve this question

Follow

edited Nov 4, 2022 at 8:18



[codeforester](#)

42.8k ● 19 ● 118 ● 153

asked Sep 3, 2009 at 3:11



[Derek Dahmer](#)

15.5k ● 6 ● 37 ● 33

-
- 2 With full path, see: [Getting the source directory of a Bash script from within](#). – [kenorb](#) Jul 19, 2017 at 18:38
-



1526



No need for `basename`, and especially no need for a subshell running `pwd` (which [adds an extra, and expensive, fork operation](#)); the shell can do this internally using [parameter expansion](#):

```
result=${PWD##*/}          # to assign to a variable
result=${result:-/}        # to correct for the case w

printf '%s\n' "${PWD##*/}" # to print to stdout
                           # ...more robust than echo
                           # (consider a directory

printf '%q\n' "${PWD##*/}" # to print to stdout, quote
                           # ...useful to make hidden
```

Note that if you're applying this technique in other circumstances (not `PWD`, but some other variable holding a directory name), you might need to trim any trailing slashes. The below uses bash's [extglob support](#) to work even with multiple trailing slashes:

```
dirname=/path/to/somewhere//
shopt -s extglob          # enable +(...) glob syntax
result=${dirname%+(/)}    # trim however many trailing
result=${result##*/}      # remove everything before
remains                   #
result=${result:-/}       # correct for dirname=/ cas
printf '%s\n' "$result"
```

Alternatively, without `extglob`:

```
dirname="/path/to/somewhere/"
result="${dirname%${dirname##*[!/]}"}" # extglob-free
result="${result##*/}"                 # remove every
result=${result:-/}                    # correct for
```

Share Improve this answer

edited Oct 12 at 1:22

Follow



Hashim Aziz

5,908 ● 5 ● 53 ● 94

answered Sep 3, 2009 at 3:21



Charles Duffy

294k ● 43 ● 430 ● 481

48 What is the difference between `${PWD##*/}` and `$PWD`?

– [Mr_Chimp](#) Nov 22, 2011 at 12:34 ✎

39 @Mr_Chimp the former is a parameter expansion operations which trims the rest of the directory to provide only the basename. I have a link in my answer to the documentation on parameter expansion; feel free to follow that if you have any questions. – [Charles Duffy](#) Nov 25, 2011 at 14:07 ✎

42 @stefgosselin `$PWD` is the name of a built-in bash variable; all built-in variable names are in all-caps (to distinguish them from local variables, which should always have at least one lower-case character). `result=${PWD##*/}` does *not* evaluate to `/full/path/to/directory`; instead, it strips only the first element, making it `path/to/directory`; using two `#` characters makes the pattern match greedy, matching as many characters as it can. Read the parameter expansion page, linked in the answer, for more.
– [Charles Duffy](#) Mar 20, 2013 at 12:20

10 Note that the output from `pwd` is not always the same as the value of `$PWD`, due to complications arising from `cd` ing through symbolic links, whether bash has the `-o physical` option set, and so on. This used to get especially

nasty around handling of automounted directories, where recording the physical path instead of the logical one would produce a path that, if used, would allow the automounter to spontaneously dismount the directory one was using.

– [Alex North-Keys](#) May 17, 2013 at 9:53

- 9 Nice! Someone should write a book for old Borne shell guys like me. Maybe there is one out there? It could have a crotchity old sys admin saying stuff like, "back in my day we only `sh` and `csh` and if you wanted the backspace key to work you had to read the whole `stty` man page, and we liked it!" – [Red Cricket](#) Oct 3, 2013 at 22:17



579



Use the `basename` program. For your case:

```
% basename "$PWD"  
bin
```

Share Improve this answer

edited Aug 17, 2018 at 12:45

Follow

answered Sep 3, 2009 at 3:13



[Arkady](#)

15.1k ● 8 ● 43 ● 47

- 30 Isn't this the purpose of the `basename` program? What is wrong with this answer besides missing the quotes? – [Nacht](#) Apr 30, 2013 at 7:18


- 21 @Nacht `basename` is indeed an external program that does the right thing, but running *any* external program for a thing bash can do out-of-the-box using only built-in functionality is silly, incurring performance impact (`fork()` , `execve()` ,

`wait()` , etc) for no reason. – [Charles Duffy](#) Jun 29, 2013 at 18:30

- 7 ...as an aside, my objections to `basename` here *don't* apply to `dirname` , because `dirname` has functionality that `${PWD##*/}` does not -- transforming a string with no slashes to `.` , for instance. Thus, while using `dirname` as an external tool has performance overhead, it also has functionality that helps to compensate for same.

– [Charles Duffy](#) Oct 20, 2014 at 12:29

- 84 Sure using `basename` is less "efficient". But it's probably *more* efficient in terms of developer productivity because it's easy to remember compared to the ugly bash syntax. So if you remember it, go for it. Otherwise, `basename` works just as well. Has anyone ever had to improve the "performance" of a bash script and make it more efficient? – [P.P](#) Dec 5, 2017 at 14:48
-

- 10 @usr, ...speaking as someone who's written bash scripts that operate over maildirs (a mailbox storage format with one file per email, as used by gmail), yes, *absolutely*, I've had real-world cause to pay attention to efficiency in scripting. We're talking on the scale of 20ms per command substitution on non-embedded hardware -- loop over 100,000 items and you're talking real time even if your code just has one of them. (Sure, there's a point where shell is no longer the right tool, but you reach that point far faster if you don't pay any attention to how well you write your code). – [Charles Duffy](#) Oct 24, 2018 at 13:34 
-



```
$ echo "${PWD##*/}"
```



Share Improve this answer

edited Feb 13, 2015 at 11:54

Follow



[gniourf_gniourf](#)

46.7k ● 9 ● 103 ● 110

answered Sep 3, 2009 at 3:23



[DigitalRoss](#)

146k ● 25 ● 252 ● 331

5 @jocap ...except that the usage of echo there, without quoting the argument, is *wrong*. If the directory name has multiple spaces, or a tab character, it'll be collapsed to a single space; if it has a wildcard character, it will be expanded. Correct usage would be `echo "${PWD##*/}"`.
– [Charles Duffy](#) Dec 11, 2012 at 3:04

17 @jocap ...also, I'm not sure that "echo" is in fact a legitimate part of the answer. The question was how to *get* the answer, not how to *print* the answer; if the goal was to assign it to a variable, for instance, `name=${PWD##*/}` would be right, and `name=$(echo "${PWD##*/}")` would be wrong (in a needless-inefficiency sense). – [Charles Duffy](#) Jan 21, 2013 at 17:16



You can use a combination of pwd and basename. E.g.

37



```
#!/bin/bash
```

```
CURRENT=`pwd`
```

```
BASENAME=`basename "$CURRENT"`
```

```
echo "$BASENAME"
```

```
exit;
```



Share Improve this answer

edited Feb 13, 2015 at 11:56

Follow



gniourf_gniourf

46.7k ● 9 ● 103 ● 110

answered Sep 3, 2009 at 3:20



mbelos

513 ● 4 ● 8

22 Please, no. The backticks create a subshell (thus, a fork operation) -- and in this case, you're using them *twice*! [As an additional, albeit stylistic quibble -- variable names should be lower-case unless you're exporting them to the environment or they're a special, reserved case]. – [Charles Duffy](#) Sep 3, 2009 at 3:26

13 and as a second style quibble backtics should be replaced by `$()`. still forks but more readable and with less escaping needed. – [Jeremy Wall](#) Sep 3, 2009 at 21:17

3 By convention, environment variables (PATH, EDITOR, SHELL, ...) and internal shell variables (BASH_VERSION, RANDOM, ...) are fully capitalized. All other variable names should be lowercase. Since variable names are case-sensitive, this convention avoids accidentally overriding environmental and internal variables. – [Rany Albeg Wein](#) Jan 21, 2016 at 4:09

4 Depends on the convention. One could argue that all shell variables are environment variables (depending on the shell), and thus all shell variables should be in all-caps. Someone else could argue based on scope - global variables are all-caps, while local variables are lowercase, and these are all globals. Yet another might argue that making variables all-uppercase adds an extra layer of contrast with the commands littering shell scripts, which are also all lower-case short but meaningful words. I may or may not /agree/ with any of those arguments, but I've seen all three in use. :) – [dannysauer](#) Jan 27, 2017 at 21:00



Use:

34

```
basename "$PWD"
```



OR



```
IFS=/  
var=($PWD)  
echo ${var[-1]}
```

Turn the Internal Filename Separator (IFS) back to space.

```
IFS=
```

There is one space after the IFS.

Share Improve this answer

Follow

edited Apr 17, 2020 at 18:42



Arsen Khachaturyan

8,280 ● 4 ● 45 ● 44

answered Jun 1, 2018 at 1:24



abc

3,531 ● 1 ● 16 ● 15

Reads well, less magic! – [Bret Weinraub](#) Dec 1, 2022 at 15:36

- 1 This is the same as [Arton Dorneles' 2016 answer](#) isn't it? IFS to slash, \$PWD to array, read last array element? Minus some IFS restoration, and failing if a path contains *, ?, or [characters – [Xen2050](#) Jun 6 at 20:52
-

@BretWeinraub, I disagree -- there's *lots* of magic here, it's just largely invisible (for example, unquoted expansions run string-splitting -- which is desired in the first case, undesired in the second one -- and globbing -- which is desired in *neither* case, and can cause wildly unexpected/surprising results if the directory name contains glob characters and either `nullglob` or `failglob` options are enabled). I'd argue that invisible magic is the worst kind: because people don't know it's there, they don't know what the gotchas and corner cases are. – [Charles Duffy](#) Jul 17 at 16:59

...if you see some syntax you don't know, you can look it up and understand what it does. When you see syntax that *looks* obvious but has hidden meanings that only experts understand, you don't know there's anything to look for until you step on one of the covered pitfalls. – [Charles Duffy](#) Jul 17 at 17:01



How about grep:

21

```
pwd | grep -o '^[^/]*$'
```



Share Improve this answer

answered Mar 3, 2016 at 19:31



Follow



[Orange](#)

235 ● 2 ● 2



2 would not recommend because it seems to get some color information whereas `pwd | xargs basename` doesn't.. probably not that important but the other answer is simpler and more consistent across environments – [davidhq](#) Sep 13, 2018 at 22:56



14

This thread is great! Here is one more flavor:

```
pwd | awk -F / '{print $NF}'
```



Share Improve this answer

answered Mar 27, 2013 at 13:32

Follow



[rodvlopes](#)

945 ● 1 ● 9 ● 19



14

```
basename $(pwd)
```

or



```
echo "$(basename $(pwd))"
```



Share Improve this answer

answered Feb 6, 2014 at 10:05

Follow



[user3278975](#)

159 ● 1 ● 2

3 Needs more quotes to be correct -- as it is, the output of `pwd` is string-split and glob-expanded before being passed to `basename`. – [Charles Duffy](#) Jul 6, 2014 at 16:49

1 Thus, `basename "$(pwd)"` -- though that's very inefficient compared to just `basename "$PWD"`, which is *itself* inefficient compared to using a parameter expansion instead of calling `basename` at all. – [Charles Duffy](#) May 13, 2020 at 22:18

love this because I will simply never remember `echo "${PWD##*/}"` haha 😂. – [HeyWatchThis](#) Oct 13, 2023 at

**11**

I like the selected answer (Charles Duffy), but be careful if you are in a symlinked dir and you want the name of the target dir. Unfortunately I don't think it can be done in a single parameter expansion expression, perhaps I'm mistaken. This should work:

```
target_PWD=$(readlink -f .)
echo ${target_PWD##*/}
```

To see this, an experiment:

```
cd foo
ln -s . bar
echo ${PWD##*/}
```

reports "bar"

DIRNAME

To show the leading directories of a path (without incurring a fork-exec of `/usr/bin/dirname`):

```
echo ${target_PWD%/*}
```

This will e.g. transform `foo/bar/baz` -> `foo/bar`

Follow

answered Apr 4, 2013 at 4:16



FDS

5,367 ● 2 ● 23 ● 13

-
- 6 Unfortunately, `readlink -f` is a GNU extension, and thus not available on the BSDs (including OS X).
– [Xiong Chiamiov](#) Oct 26, 2013 at 5:32
-



8



If you are using Bourne shell or `${PWD##*/}` is not available.



Share Improve this answer

edited Feb 5, 2014 at 1:10

Follow



answered May 30, 2013 at 23:46



anomal

2,289 ● 2 ● 19 ● 18

-
- 5 FYI, `${PWD##*/}` is POSIX sh -- every modern `/bin/sh` (including dash, ash, etc) supports it; to hit actual Bourne on a recent box, you'd need to be on a mildly oldish Solaris system. Beyond that -- `echo "$PWD"`; leaving out the quotes leads to bugs (if the directory name has spaces, wildcard characters, etc). – [Charles Duffy](#) Feb 3, 2014 at 14:06
-

- 3 Yes, I was using an oldish Solaris system. I have updated the command to use quotes. – [anomal](#) Feb 5, 2014 at 1:11
-



Just run the following command line:

8

```
basename $(pwd)
```



If you want to copy that name:



```
basename $(pwd) | xclip -selection clipboard
```



```
ahmnouira@pc:~/Desktop/works/flutter/enhanced_counter_app$ pwd
/home/ahmnouira/Desktop/works/flutter/enhanced_counter_app
ahmnouira@pc:~/Desktop/works/flutter/enhanced_counter_app$ basename $(pwd)
enhanced_counter_app
ahmnouira@pc:~/Desktop/works/flutter/enhanced_counter_app$ basename $(pwd) | xclip -selection clipboard
ahmnouira@pc:~/Desktop/works/flutter/enhanced_counter_app$
```

Share Improve this answer

answered Nov 3, 2022 at 10:49

Follow



[ahmnouira](#)

3,281 ● 1 ● 32 ● 16



Surprisingly, no one mentioned this alternative that uses only built-in bash commands:

7

```
i="$IFS";IFS=' / ';set -f;p=($PWD);set +f;IFS="$i";echo
```



As an **added bonus** you can easily obtain the name of the parent directory with:



```
[ "${#p[@]}" -gt 1 ] && echo "${p[-2]}"
```

These will work on Bash 4.3-alpha or newer.

Share Improve this answer

edited Mar 3, 2016 at 21:14

Follow

answered Mar 3, 2016 at 20:26



[Arton Dorneles](#)

1,709 ● 15 ● 18

surprisingly ? just joking, but others are much shorter and easier to remember – [codewandler](#) Jul 8, 2016 at 15:20

This is pretty funny =P Had not heard of \$IFS (internal field separator) before this. my bash was too old, but can test it here: jdoodle.com/test-bash-shell-script-online
– [Stan Kurdziel](#) Feb 12, 2017 at 5:50 ✎



6

There are a lots way of doing that I particularly liked [Charles](#) way because it avoid a new process, but before know this I solved it with awk



```
pwd | awk -F/ '{print $NF}'
```



Share Improve this answer

answered Aug 2, 2020 at 5:11

Follow



[geckos](#)

6,239 ● 1 ● 47 ● 62



i usually use this in sh scripts

5



```
SCRIPTSRC=`readlink -f "$0" || echo "$0"`  
RUN_PATH=`dirname "${SCRIPTSRC}" || echo .`  
echo "Running from ${RUN_PATH}"  
...  
cd ${RUN_PATH}/subfolder
```



you can use this to automate things ...

Share Improve this answer

answered Oct 1, 2016 at 16:52

Follow



[Dutch Glory](#)

26.2k ● 1 ● 19 ● 6

```
readlink: illegal option -- f usage: readlink [-  
n] [file ...] – user5549921 Jul 19, 2017 at 8:52
```



For the find jockeys out there like me:

5



```
find $PWD -maxdepth 0 -printf "%f\n"
```



Share Improve this answer

answered Apr 10, 2017 at 18:13

Follow



[user2208522](#)

51 ● 1 ● 2

2 Change the `$PWD` to `"$PWD"` to correctly handle unusual directory names. – [Charles Duffy](#) May 13, 2020 at 22:16



Just use:

4

```
pwd | xargs basename
```



or



```
basename "`pwd`"
```



Share Improve this answer

edited Feb 13, 2015 at 11:40

Follow



fedorqui

289k ● 108 ● 585 ● 627

answered Feb 13, 2015 at 11:34



marcos

65 ● 1

-
- 3 This is in all respects a worse version of the answer previously given by Arkady (stackoverflow.com/a/1371267/14122): The `xargs` formulation is inefficient and buggy when directory names contain literal newlines or quote characters, and the second formulation calls the `pwd` command in a subshell, rather than retrieving the same result via a built-in variable expansion. – Charles Duffy Jun 8, 2015 at 0:55

@CharlesDuffy Nevertheless is it a valid and practical answer to the question. – bachph Mar 14, 2020 at 9:53

@bachph, I disagree: A buggy answer (f/e, an answer that doesn't work when directory names contain spaces) *should not be considered an answer at all*. – Charles Duffy Jun 6, 2022 at 14:49



3

Below grep with regex is also working,

```
>pwd | grep -o "\w*-*$"
```



Share Improve this answer

edited Apr 6, 2017 at 3:03



Follow



answered Mar 28, 2017 at 11:20



Abhishek Gurjar

7,476 ● 10 ● 40 ● 47

4 It doesn't work if directory name contains "-" characters.
– [daniula](#) Apr 5, 2017 at 19:40



3

If you want to see only the current directory in the bash prompt region, you can edit `.bashrc` file in `~`. Change `\w` to `\W` in the line:



```
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]  
[\033[01;34m\]\W\[\033[00m\]\$ '
```



Run `source ~/.bashrc` and it will only display the directory name in the prompt region.

Ref: <https://superuser.com/questions/60555/show-only-current-directory-name-not-full-path-on-bash-prompt>

answered Oct 16, 2019 at 10:44

Share Improve this answer

Follow



Gautam Sreekumar

554 ● 2 ● 9 ● 22



I strongly prefer using `gbasename`, which is part of GNU coreutils.

2



Share Improve this answer

answered Jul 27, 2017 at 15:19

Follow



Steve Benner

1,707 ● 22 ● 26



2 FYI, it doesn't come with my Ubuntu distribution.

– [Katastic Voyage](#) Sep 25, 2017 at 8:34

2 @KatasticVoyage, it's there on Ubuntu, it's just called `basename` there. It's only typically called `gbasename` on MacOS and other platforms that otherwise ship with a non-GNU `basename`. – [Charles Duffy](#) May 13, 2020 at 22:15 ✎



Here's a simple alias for it:

2



```
alias name='basename $( pwd )'
```



After putting that in your `~/.zshrc` or `~/.bashrc` file and sourcing it (ex: `source ~/.zshrc`), then you can simply run `name` to print out the current directories name.

Share Improve this answer

answered Feb 16, 2023 at 16:45

Follow



Praxder

2,729 ● 4 ● 37 ● 54



An alternative to `basename` examples

1

```
pwd | grep -o "[^/]*$"
```



OR



```
pwd | ack -o "[^/]+$"
```



My shell did not come with the `basename` package and I tend to avoid downloading packages if there are ways around it.

Share Improve this answer

answered Nov 29, 2022 at 19:16

Follow



D3MON

41 ● 4



You can use the `basename` utility which deletes any prefix ending in `/` and the suffix (if present in string) from string, and prints the result on the standard output.

0



```
$basename <path-of-directory>
```



Share Improve this answer

answered Oct 3, 2013 at 22:05

Follow



niks_4143

97 ● 1 ● 8



0

Just remove any character until a `/` (or `\`, if you're on Windows). As the match is gonna be made greedy it will remove everything until the last `/`:



```
pwd | sed 's/.*\///g'
```



In your case the result is as expected:

```
λ a='/opt/local/bin'

λ echo $a | sed 's/.*\///g'
bin
```

Share Improve this answer

answered Jul 28, 2021 at 10:15

Follow



Benjamin Zach

1,601 ● 2 ● 21 ● 43



-1

The following commands will result in printing your current working directory in a bash script.



```
pushd .
CURRENT_DIR="`cd $1; pwd`"
popd
echo $CURRENT_DIR
```



Share Improve this answer

edited May 17, 2012 at 15:22

Follow



octopusgrabbus

10.7k ● 15 ● 74 ● 135

answered May 16, 2012 at 13:56



Paul Sabou

3,197 ● 3 ● 17 ● 9

-
- 3 (1) I'm not sure where we're ensuring that the argument list is such that `$1` will contain the current working directory. (2) The `pushd` and `popd` serve no purpose here because anything inside backticks is done in a subshell -- so it can't affect the parent shell's directory to start with. (3) Using `"$(cd "$1"; pwd)"` would be both more readable and resilient against directory names with whitespace.
- [Charles Duffy](#) Jun 13, 2012 at 16:21
-



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.