How to export variables that are set, all at once?

Asked 9 years, 8 months ago Modified 18 days ago Viewed 57k times

78

`set` command displays all the local variables like below. How do I export these variables all at once?

```
>set
a=123
b="asd asd"
c="hello world"
```

bash | environment-variables | set

Share Improve this question Follow

edited Jan 23 at 10:38
Kusalananda ♦
307k 35 599 896

asked Jun 11, 2013 at 20:23
Neerav
2,775 3 15 7

what do you mean export all at once? you can use semi colons to define in one line...like a=123;b="asd asd";c="hello world" – Raza Jun 11, 2013 at 20:57

Very similar to unix.stackexchange.com/q/79064/4667 – glenn jackman Jun 11, 2013 at 21:00

2 `set` also displays functions and system variables like BASH_VERSION – glenn jackman Jun 11, 2013 at 21:02

Your question is unclear. Is that an excerpt of `set` output you're showing? If so, then it's not `bash` 's. Do you want to export all the currently set variable including the special shell variables? Or only those 3 variables like in `export a b c` ? – Stéphane Chazelas Jun 11, 2013 at 21:07

`export ${!T*}` would export any defined parameter whose name starts with `T` . Unfortunately, there doesn't seem to be a way to easily generate a list of *all* defined parameters. – chepner Jun 13, 2013 at 15:43

5 Answers

Sorted by: Highest score (default)

137

Run the following command, before setting the variables:

```
set -a
set -o allexport  # self-documenting version
```

man page:

> `-a`
> When this option is on, the export attribute shall be set for each variable to which an assignment is

performed

`-o option-name`

Set the option corresponding to `option-name` :

- `allexport`
  Same as `-a` .

To turn this option off, run `set +a` or `set +o allexport` afterwards.

Example:

```
set -a  # or: set -o allexport
. ./environment
set +a
```

Where `environment` contains:

```
FOO=BAR
BAS='quote when using spaces, (, >, $, ; etc'
```

Share Improve this answer Follow

edited Jan 23 at 8:35 — Stéphane Chazelas — 498k 90 967 1445

answered Jun 11, 2013 at 22:09 — Nitin4873 — 1,551 1 10 5

17 This must be enabled before assigning to variables, though. It doesn't do anything to previously assigned variables. – chepner Jun 13, 2013 at 15:13

1 @chepner , Thanks i forgot to mention that !! – Nitin4873 Jun 13, 2013 at 15:41

2 This also automatically exports functions in the same way as `function example(){ echo good; }; export -f example` – Oliver I Aug 29, 2017 at 15:16

8

Same preliminary requirement as chosen answer ... either explicitly export each variable as per

```
export aaaa=1234
```

or prior to any variable assignment issue

```
set -a  #  for details see answer by @nitin
```

then this works if your shell is bash (possibly other shells as well)

```
export > /my/env/var/file
```

your new file will contain a dump of all currently defined variables ... with entries like

```
declare -x PORT="9000"
declare -x PORT_ADMIN="3001"
declare -x PORT_DOCKER_REGISTRY="5000"
declare -x PORT_ENDUSER="3000"
declare -x
PRE_BUILD_DIR="/cryptdata6/var/log/tmp/khufu01/loud_deploy/curr/loud-
build/hygge"
declare -x PROJECT_ID="hygge"
declare -x PROJECT_ID_BUSHIDO="bushido"
```

then to jack up current shell with all those env vars issue

```
source  /my/env/var/file
```

Share Improve this answer Follow

edited Mar 10, 2022 at 12:07

answered Apr 3, 2018 at 16:16

Scott Stensland
2,493 2 23 23

I believe `export` only prints variables that are already (marked to be) exported, while I believe the question is about variables that are set, but not marked to be exported. I just tested this on bash, which prints nothing: `FOOBAR=x; export | grep FOOBAR` . – Matthijs Kooijman Mar 9, 2022 at 15:58

@MatthijsKooijman good catch I updated to qualify my answer – Scott Stensland Mar 10, 2022 at 12:07

5

```
`echo "export" $((set -o posix ; set)|awk -F "=" 'BEGIN{ORS=" "}1 $1~/[a-zA-Z_]
[a-zA-Z0-9_]*/ {print $1}')`
```

1. First, get all set environment variables: `(set -o posix ; set)` Reference: https://superuser.com/questions/420295/how-do-i-see-a-list-of-all-currently-defined-environment-variables-in-a-linux-ba
2. Get all environment variable names, separated by space: `awk -F "=" 'BEGIN{ORS=" "}1 $1~/[a-zA-Z_][a-zA-Z0-9_]*/ {print $1}'` Reference: awk-Printing column value without new line and adding comma and https://stackoverflow.com/questions/14212993/regular-expression-to-match-a-pattern-inside-awk-command
3. Now, we need to export these variables, but xargs can not do this because it forks child process, export have to be run under current process. `echo "export" ...` build a command we want, then use `` to run it. That's all :p.

Share Improve this answer Follow

edited Aug 29, 2017 at 17:10

Wil Moore III
139 4

answered Mar 6, 2015 at 10:16

Bear Huang
51 1 2

Welcomme to U&L SE. Maybe you can edit your post and give some explication. – Archemar Mar 6, 2015 at 12:09

1 It's not correct to assume environment variable names will consist solely of a-z and A-Z. They commonly include underscores and digits as well, so the pattern would be `[a-zA-Z_][a-zA-Z0-9_]*` . There are some variations on this based on the shell you're using, but this is the safe / portable approach. – Chris Johnson May 5, 2017 at 17:27

Good point @ChrisJohnson - updated! – Wil Moore III Aug 29, 2017 at 15:52

2

You can prepend `export` to the variable name via `awk` and `eval` the resulting output:

```
eval $(printenv | awk -F= '{ print "export " $1 }')
```

Share Improve this answer Follow

edited Nov 16, 2016 at 14:00

muru

67.3k 11 187 281

answered Dec 30, 2015 at 2:49

Wil Moore III

139 4

3 `printenv` prints the variables that are already exported. That also won't work properly if there are variables that contain newline characters. – Stéphane Chazelas Aug 29, 2017 at 10:57

2

`compgen -v` will print a list of all variable names so you can export them all with

```
export $(compgen -v)
```

This will have various effects depending on the variables you have defined (ex: `BASHOPTS` will get exported by this). Be wary of how you use this.

Share Improve this answer Follow

edited Feb 12, 2022 at 0:53

answered Feb 6, 2022 at 10:37

johncs

21 3

Or just `export $(compgen -v)` (also assuming an unmodified `$IFS`), bearing in mind that exporting bash builtin variables such as `SHELLOPTS` or `BASHOPTS` can have nasty consequences. – Stéphane Chazelas Feb 6, 2022 at 15:01

Thanks, I never realized `export` accepts multiple symbols. I've incorporated most of your comment into the answer. – johncs Feb 12, 2022 at 0:54