set

Change the value of a shell option and set the positional parameters, or display the names and values of shell variables.

Syntax

```
      set [--abefhkmnptuvxBCEHPT] [-o option-name] [argument …]
      set [+abefhkmnptuvxBCEHPT] [+o option-name] [argument …]
```

If no options or arguments are supplied, set displays the names and values of all shell variables and functions, sorted according to the current locale, in a format that can be reused as input. When options are supplied, they set or unset shell attributes.

Options

Using + rather than - will cause the option to be turned off.

-a Mark variables which are modified or created for export. **-o allexport**

-b Cause the status of terminated background jobs to be
    reported immediately, rather than before printing the
    next primary prompt. **-o notify**

-B The shell will perform brace expansion.
    This option is on by default. **-o braceexpand**

-C Prevent output redirection using '>', '>&',
    and '<>' from overwriting existing files. **-o noclobber**

-e Exit immediately if a simple command exits with a non-zero status, unless
    the command that fails is part of an until or while loop, part of an
    if statement, part of a && or || list, or if the command's return status
    is being inverted using !. **-o errexit**

-f Disable file name generation (globbing). **-o noglob**

-h Locate and remember (hash) commands as they are looked
    up for execution. This option is enabled by default. **-o hashall**

-H Enable '!' style history substitution
    This option is on by default for interactive shells. **-o histexpand**

-k All arguments in the form of assignment statements are
    placed in the environment for a command, not just those that
    precede the command name. **-o keyword**

-m Job control is enabled. **-o monitor**

-n Read commands but do not execute them; this can be used
    to check a script for syntax errors.
    This option is ignored by interactive shells. **-o noexec**

-o If -o is supplied with no *option-name*, the values of the current options are printed.
    If +o is supplied with no *option-name*, a series of set commands to recreate the current
    option settings is displayed on the standard output.

-o *option-name*
       Set the option corresponding to '*option-name*'
       The 'option-names' are listed above and below (in **bold**):

```
              allexport   Same as -a.
              braceexpand Same as -B.
              emacs       Use an emacs-style command line editing interface.  This is enabled by defaul
                          when the shell is interactive, unless the shell is started with --noediting
              errtrace    Same as -E.
              functrace   Same as -T.
              errexit     Same as -e.
              hashall     Same as -h.
              histexpand  Same as -H.
              history     Enable command history, as described above under HISTORY.  This option  is  o
                           by default in interactive shells.
              ignoreeof   The  effect is as if the shell command 'IGNOREEOF=10' had been executed
              keyword     Same as -k.
              monitor     Same as -m.
              noclobber   Same as -C.
              noexec      Same as -n.
              noglob      Same as -f.  nolog Currently ignored.
              notify      Same as -b.
              nounset     Same as -u.
              onecmd      Same as -t.
              physical    Same as -P.
              pipefail    If set, the return value of a pipeline is the value of the last (rightmost) c
                          exit with a non-zero status, or zero if all commands in the pipeline exit suc
```

```
                            By default, pipelines only return a failure if the last command errors.

                            When used in combination with set -e, pipefail will make a script exit if any
                            in a pipeline errors.

                    posix       Change the behavior of bash where the default operation differs from the POSI
                                standard to match the standard (posix mode).
                    privileged  Same as -p.
                    verbose     Same as -v.
                    vi          Use a vi-style command line editing interface.
                    xtrace  Same as -x.

    -p  Turn on privileged mode.
        In this mode, the $BASH_ENV and $ENV files are not processed, shell functions are not
        inherited from the environment,  and the SHELLOPTS variable, if it appears in the environment,
        is ignored. If the shell is started with the effective user (group) id not equal to the
        real user (group) id, and the -p option is not supplied, these actions are taken and
        the effective user id is set to the real user id.
        If the -p option is supplied at startup, the effective user id is not reset.

        Turning this option off causes the effective user and group ids to be set to the real user
        and group ids.   -o privileged

    -P  If set, do not follow symbolic links when performing commands.
        The physical directory is used instead. -o physical

    -t  Exit after reading and executing one command.  -o onecmd

    -u  Treat unset variables as an error when performing
        parameter expansion. An error message will be written
        to the standard error, and a non-interactive shell will exit. -o nounset

    -v  Print shell input lines as they are read. -o verbose

    -x  Print a trace of simple commands and their arguments
        after they are expanded and before they are executed. -o xtrace

    --  If no arguments follow this option, then the positional parameters are unset.
        Otherwise, the positional parameters are set to the arguments,
        even if some of them begin with a '-'.

    -   Signal the end of options, cause all remaining arguments to be
        assigned to the positional parameters. The '-x' and '-v'
        options are turned off. If there are no arguments, the positional parameters
        remain unchanged.

Without options, the name and value of each shell variable are displayed in a format that can be reused as input for setting
or resetting the currently-set variables.

When options are specified, they set or unset shell attributes.

Read-only variables cannot be reset.

By default, unset variables are evaluated as an empty string which can have unexpected and undesirable effects.

When writing and debugging shell scripts it is good practice to set both -e and -u so that the script will exit on an Error
or if an Unset variable is referenced : set -eu

The options can also be specified as arguments to an invocation of the shell. The current set of options may be found in $-.
Any arguments remaining after the options are processed are treated as values for the positional parameters and are assigned,
in order, to $1, $2, ... $N. The special parameter # is set to N.
```

Symbolic Links

By default, Bash follows the logical chain of directories when performing commands which change the current directory. e.g.

If '/usr/sys' is a symbolic link to '/usr/local/sys' then:

```
$ cd /usr/sys; echo $PWD
/usr/sys
$ cd ..; pwd
/usr
```

If **set -P** is on (do not follow symbolic links), then:

```
$ cd /usr/sys; echo $PWD
/usr/local/sys
$ cd ..; pwd
/usr/local
```

In posix mode, only shell variables are listed.
The output is sorted according to the current locale.
The return status is always zero unless an invalid option is supplied.
This is a BASH shell builtin, to display your local syntax from the bash prompt type: help set

Examples

Set both -e and -u so that the script will exit on an **E**rror or if an **U**nset variable is referenced. This is good practice when writing and debugging shell scripts.

```
#!/bin/bash
# an example bash script
set -eu
```

Set pipefail so that any failure in a pipeline will return an error:

```
#!/bin/bash
set -o pipefail
```

Set both -e and -u and pipefail:

```
#!/bin/bash
set -eu -o pipefail
```

Set all of the above for maximum error trapping within a script:

```
#!/bin/bash
set -euo pipefail
```

Turn ON the allexport option:

```
set -o allexport
```

Turn OFF the allexport option:

```
set +o allexport
```

Set the shell variable 'MYDEPT' equal to 'Sales':

```
MYDEPT=Sales
echo $MYDEPT
```

To make the change permanent, you can export this as an environment variable:

```
export $MYDEPT
```

Debug part of a script:

```
set -x  # Activate debugging
# your commands go here...
set +x  # stop debugging
```

Disable strict mode while you source a file:

```
set +u  # Disable variable strictness
source ss64file.env
set -u  # re-enable variable strictness
```

"I can understand there are things like shadows they need to fix after a shoot, but it's unfair to represent an image of yourself if it's not true. They're gonna see what you look like on film anyway, so why try to cover all your wobbly bits in a photo?" ~ Emily Blunt

Related linux commands

env - Display, set, or remove environment variables.
export - Set an *environment* variable.
groups - Print group names a user is in.
hostname - Print or set system name.
id - Print user and group id's.
logname - Print current login name.
printenv - Print environment variables.
readonly - Mark variables/functions as readonly.
Bash shell variables
shift - Shift positional parameters.
shopt - Shell Options.
uname - Print system information.
users - Print login names of users currently logged in.
unset - Remove variable or function names.
who - Print who is currently logged in.
Equivalent Windows commands: SET - Display, set, or remove Windows environment variables.