

Special parameters and shell variables

Special Parameters

parameter	character	expansion description
*	asterisk	<p>The positional parameters starting from the first. When used inside doublequotes (see quoting), like "\$*", it expands to all positional parameters as <i>one word</i>, delimited by the first character of the IFS variable (a space in this example): "\$1 \$2 \$3 \$4" .</p> <p>If IFS is unset, the delimiter used will be always a space, if IFS is NULL, the delimiter will be nothing, which effectively concatenates all the positional parameters without any delimiter.</p> <p>When used unquoted, it will just expand to the strings, one by one, not preserving the word boundaries (i.e. word splitting will split the text again, if it contains IFS characters.</p> <p>See also the scripting article about handling positional parameters.</p>
@	at-sign	<p>The positional parameters starting from the first. When used inside doublequotes (see quoting), like "\$@" , it expands all positional parameters as <i>separate words</i>: "\$1" "\$2" "\$3" "\$4"</p> <p>Without doublequotes, the behaviour is like the one of * without doublequotes.</p> <p>See also the scripting article about handling positional parameters.</p>
#	hash mark	<p>Number of positional parameters (decimal)</p> <p>See also the scripting article about handling positional parameters.</p>
?	question mark	<p>Status of the most recently executed foreground-pipeline (exit/return code)</p>
-	dash	<p>Current option flags set by the shell itself, on invocation, or using the set builtin command. It's just a set of characters, like himB for h , i , m and B .</p>

parameter	character	expansion description
\$	dollar-sign	The process ID (PID) of the shell. In an explicit subshell it expands to the PID of the current "main shell", not the subshell. This is different from \$BASHPID !
!	exclamation mark	The process ID (PID) of the most recently executed background pipeline (like started with command &)
0	zero	<p>The name of the shell or the shell script (filename). Set by the shell itself.</p> <p>If Bash is started with a filename to execute (script), it's set to this filename. If started with the -c <CMDLINE> option (commandline given as argument), then \$0 will be the first argument after the given <CMDLINE> . Otherwise, it is set to the string given on invocation for argv[0] .</p> <p>Unlike popular belief, \$0 is <i>not a positional parameter</i>.</p>
_	underscore	<p>A kind of catch-all parameter. Directly after shell invocation, it's set to the filename used to invoke Bash, or the absolute or relative path to the script, just like \$0 would show it.</p> <p>Subsequently, expands to the last argument to the previous command. Placed into the environment when executing commands, and set to the full pathname of these commands. When checking mail, this parameter holds the name of the mail file currently being checked.</p>

Shell Variables

BASH

Variable:	BASH	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Expands to the full file name used to invoke the current instance of Bash.

BASHOPTS

Variable:	BASHOPTS	Since:	4.1-alpha
Type:	normal variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

A colon-separated list of enabled shell options.

Each word in the list is a valid argument for the `-s` option to the `shopt` builtin command. The options appearing in `BASHOPTS` are those reported as on by `shopt`. If this variable is in the environment when Bash starts up, each shell option in the list will be enabled before reading any startup files.

Example content:

```
cmdhist:expand_aliases:extquote:force_fignore:hostcomplete:interactiv
e_comments:progcomp:promptvars:sourcepath
```

This variable is read-only.

BASHPID

Variable:	BASHPID	Since:	4.0-alpha
Type:	integer variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

Always expands to the process ID of the current Bash process. This differs from the special parameter `$` under certain circumstances, such as subshells that do not require Bash to be re-initialized.

BASH_ALIASES

Variable:	BASH_ALIASES	Since:	unknown
Type:	associative array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An associative array variable whose members correspond to the internal list of aliases as maintained by the `alias` builtin. Elements added to this array appear in the alias list; unsetting array elements cause aliases to be removed from the alias list.

The associative key is the name of the alias as used with the `alias` builtin command.

BASH_ARGC

Variable:	BASH_ARGC	Since:	3.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	only in extended debugging mode	Default:	n/a

An array variable whose values are the number of parameters in each frame of the current Bash execution call stack.

The number of parameters to the current subroutine (shell function or script executed with "." or "source" builtin command) is at the top of the stack. When a subroutine is executed, the number of parameters passed is pushed onto `BASH_ARGC` .

BASH_ARGV

Variable:	BASH_ARGV	Since:	3.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	only in extended debugging mode	Default:	n/a

An array variable containing all of the parameters in the current Bash execution call stack.

The final parameter of the last subroutine call is at the top of the stack; the first parameter of the initial call is at the bottom. When a subroutine is executed, the parameters supplied are pushed onto `BASH_ARGV` .

BASH_ARGV0

Variable:	BASH_ARGV0	Since:	5.0-alpha
Type:	string	Read-only:	no
Set by Bash:	yes	Default:	same as <code>\$0</code>

Expands to the name of the shell or shell script - as the special parameter `$0` does. Assignments to `BASH_ARGV0` causes the value to be assigned to `$0` .

If this parameter is unset, it loses its special properties, even if subsequently reset.

BASH_CMDS

Variable:	BASH_CMDS	Since:	unknown
Type:	associative array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An associative array variable whose members correspond to the internal hash table of commands as maintained by the hash builtin command. Elements added to this array appear in the hash table; unsetting array elements cause commands to be removed from the hash table.

The associative key is the name of the command as used with the hash builtin command.

BASH_COMMAND

Variable:	BASH_COMMAND	Since:	3.0
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The command currently being executed or about to be executed, unless the shell is executing a command as the result of a trap, in which case it is the command executing at the time of the trap.

BASH_COMPAT

Variable:	BASH_COMPAT	Since:	4.3-alpha
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The value is used to set the shell's compatibility level. The value may be a decimal number (e.g., 4.2) or an integer (e.g., 42) corresponding to the desired compatibility level. If BASH_COMPAT is unset or set to the empty string, the compatibility level is set to the default for the current version. If BASH_COMPAT is set to a value that is not one of the valid compatibility levels, the shell prints an error message and sets the compatibility level to the default for the current version. The valid compatibility levels correspond to the compatibility options accepted by the shopt builtin. The current version is also a valid value.

BASH_EXECUTION_STRING

Variable:	BASH_EXECUTION_STRING	Since:	3.0
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The command argument to the -c invocation option.

BASH_LINENO

Variable:	BASH_LINENO	Since:	3.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable whose members are the line numbers in source files corresponding to each member of FUNCNAME .

`${BASH_LINENO[$i]}` is the line number in the source file where `${FUNCNAME[$i]}` was called. The corresponding source file name is `${BASH_SOURCE[$i]}`. Use `LINENO` to obtain the current line number.

BASH_REMATCH

Variable:	BASH_REMATCH	Since:	3.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable whose members are assigned by the `=~` binary operator to the `[[` conditional command.

The element with index 0 is the portion of the string matching the entire regular expression. The element with index `n` is the portion of the string matching the `n`th parenthesized subexpression.

Before Bash version 5.1-alpha this variable was readonly.

BASH_SOURCE

Variable:	BASH_SOURCE	Since:	3.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable whose members are the source filenames corresponding to the elements in the `FUNCNAME` array variable.

BASH_SUBSHELL

Variable:	BASH_SUBSHELL	Since:	3.0
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Incremented by one each time a subshell or subshell environment is spawned. The initial value is 0.

BASH_VERSIONINFO

Variable:	BASH_VERSIONINFO	Since:	2.0
Type:	integer indexed array	Read-only:	yes

Set by Bash:	yes	Default:	n/a
---------------------	-----	-----------------	-----

A readonly array variable whose members hold version information for this instance of Bash. The values assigned to the array members are as follows:

BASH_VERSION[0]	The major version number (the release)
BASH_VERSION[1]	The minor version number (the version)
BASH_VERSION[2]	The patch level
BASH_VERSION[3]	The build version
BASH_VERSION[4]	The release status (e.g., beta1)
BASH_VERSION[5]	The value of <code>MACHTYPE</code>

BASH_VERSION

Variable:	BASH_VERSION	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Expands to a string describing the version of this instance of Bash.

Since Bash 2.0 it includes the shell's "release status" (`alpha[N]`, `beta[N]`, `release`).

CHILD_MAX

Variable:	CHILD_MAX	Since:	4.3-alpha
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

Set the number of exited child status values for the shell to remember. Bash will not allow this value to be decreased below a POSIX-mandated minimum, and there is a maximum value (currently 8192) that this may not exceed. The minimum value is system-dependent.

COMP_CWORD

Variable:	COMP_CWORD	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

An index into `COMP_WORDS` of the word containing the current cursor position.

COMP_KEY

Variable:	COMP_KEY	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

The key (or final key of a key sequence) used to invoke the current completion function.

COMP_LINE

Variable:	COMP_LINE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

The current command line.

COMP_POINT

Variable:	COMP_POINT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

The index of the current cursor position relative to the beginning of the current command. If the current cursor position is at the end of the current command, the value of this variable is equal to `${#COMP_LINE}` .


COMP_TYPE

Variable:	COMP_TYPET	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

Set to an integer value corresponding to the type of completion attempted that caused a completion function to be called:

TAB	normal completion
?	listing completions after successive tabs
!	listing alternatives on partial word completion

@	to list completions if the word is not unmodified
%	for menu completion

 where are the integer values?

COMP_WORDBREAKS

Variable:	COMP_WORDBREAKS	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Reports the set of characters that the readline library treats as word separators when performing word completion.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

COMP_WORDS

Variable:	COMP_WORDS	Since:	unknown
Type:	integer indexed array	Read-only:	no
Set by Bash:	only for programmable completion facilities	Default:	n/a

An array variable consisting of the individual words in the current command line. The line is split into words as readline would split it, using COMP_WORDBREAKS as described above.

COPROC

Variable:	COPROC	Since:	unknown
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable created to hold the file descriptors for output from and input to an unnamed coprocess.

DIRSTACK

Variable:	DIRSTACK	Since:	unknown
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable containing the current contents of the directory stack.

Directories appear in the stack in the order they are displayed by the `dirs` builtin. Assigning to members of this array variable may be used to modify directories already in the stack, but the `pushd` and `popd` builtins must be used to add and remove directories.

Assignment to this variable will not change the current directory.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

EPOCHREALTIME

Variable:	EPOCHREALTIME	Since:	5.0-alpha
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Expands to the number of seconds since Unix epoch as a floating point value with micro-second granularity.

Assignments to this parameter are ignored. If this parameter is unset, it loses its special properties, even if it is subsequently reset.

EPOCHSECONDS

Variable:	EPOCHSECONDS	Since:	5.0-alpha
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Expands to the number of seconds since Unix epoch.

Assignments to this parameter are ignored. If this parameter is unset, it loses its special properties, even if it is subsequently reset.

EUID

Variable:	EUID	Since:	unknown
Type:	integer variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

Expands to the effective user ID of the current user, initialized at shell startup.

⚠ Do not rely on this variable when security is a concern.

FUNCNAME

Variable:	<code>FUNCNAME</code>	Since:	2.04
Type:	integer indexed array	Read-only:	no
Set by Bash:	only inside shell functions	Default:	n/a

An array variable containing the names of all shell functions currently in the execution call stack.

The element with index 0 is the name of any currently-executing shell function. The bottom-most element (the one with the highest index) is "main".

This variable can be used with `BASH_LINENO` and `BASH_SOURCE` : Each element of `FUNCNAME` has corresponding elements in `BASH_LINENO` and `BASH_SOURCE` to describe the call stack. For instance, `${FUNCNAME[$i]}` was called from the file `${BASH_SOURCE[$i+1]}` at line number `${BASH_LINENO[$i]}` . The caller builtin command displays the current call stack using this information.

This variable exists only when a shell function is executing.

Assignments to this parameter have no effect and return an error status.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

GROUPS

Variable:	<code>GROUPS</code>	Since:	2.01
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable containing the list of groups of which the current user is a member.

Assignments to this parameter have no effect and return an error status.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

HISTCMD

Variable:	<code>HISTCMD</code>	Since:	1.14.0
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Expands to the history number (index in the history list) of the current command.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

HOSTNAME

Variable:	HOSTNAME	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Automatically set to the name of the current host.

HOSTTYPE

Variable:	HOSTTYPE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	system-dependent

Automatically set to a string that uniquely describes the type of machine on which Bash is executing.

Example content:

```
x86_64
```

LINENO

Variable:	LINENO	Since:	unknown
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Each time this parameter is referenced, the shell substitutes a decimal number representing the current sequential line number (starting with 1) within a script or function.

When not in a script or function, the value substituted is not guaranteed to be meaningful.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

MACHTYPE

Variable:	MACHTYPE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	system-dependent

Automatically set to a string that fully describes the system type on which Bash is executing, in the standard GNU "cpu-company-system" format.

Example content:

```
x86_64-unknown-linux-gnu
```

MAPFILE

Variable:	MAPFILE	Since:	unknown
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable created to hold the text read by the mapfile builtin command when no variable name is supplied.

OLDPWD

Variable:	OLDPWD	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The previous working directory as set by the cd command.

OPTARG

Variable:	OPTARG	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The value of the last option argument processed by the getopt builtin command.

OPTIND

Variable:	OPTIND	Since:	unknown
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The index of the next argument to be processed by the getopt builtin command.

OSTYPE

Variable:	OSTYPE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	system-dependent

Automatically set to a string that describes the operating system on which Bash is executing.

Example content:

```
linux-gnu
```

PIPESTATUS

Variable:	PIPESTATUS	Since:	2.0
Type:	integer indexed array	Read-only:	no
Set by Bash:	yes	Default:	n/a

An array variable containing a list of exit status values from the processes in the most-recently-executed foreground pipeline (which may contain only a single command).

PPID

Variable:	PPID	Since:	unknown
Type:	integer variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

The process ID of the shell's parent process.

PWD

Variable:	PWD	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The current working directory as set by the cd builtin command.

RANDOM

Variable:	RANDOM	Since:	unknown
------------------	--------	---------------	---------

Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Each time this parameter is referenced, a random integer between 0 and 32767 is generated. The sequence of random numbers may be initialized by assigning a value to `RANDOM` .

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

READLINE_LINE

Variable:	READLINE_LINE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The contents of the readline line buffer, for use with `bind -x` .

READLINE_POINT

Variable:	READLINE_POINT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

The position of the insertion point in the readline line buffer, for use with `bind -x` .

REPLY

Variable:	REPLY	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	only by the read builtin command	Default:	n/a

Set to the line of input read by the read builtin command when no arguments are supplied that name target variables.

SECONDS

Variable:	SECONDS	Since:	unknown
Type:	integer variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Each time this parameter is referenced, the number of seconds since shell invocation is returned. If a value is assigned to SECONDS, the value returned upon subsequent references is the number of seconds since the assignment plus the value assigned.

If this parameter is unset, it loses its special properties, even if it is subsequently reset.

SHELLOPTS

Variable:	SHELLOPTS	Since:	unknown
Type:	normal variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

A colon-separated list of enabled shell options. Each word in the list is a valid argument for the `-o` option to the `set` builtin command. The options appearing in `SHELLOPTS` are those reported as on by `set -o`.

If this variable is in the environment when Bash starts up, each shell option in the list will be enabled before reading any startup files.

SHLVL

Variable:	SHLVL	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	n/a

Incremented by one each time an instance of Bash is started.

UID

Variable:	UID	Since:	unknown
Type:	integer variable	Read-only:	yes
Set by Bash:	yes	Default:	n/a

Expands to the user ID of the current user, initialized at shell startup.

⚠ Do not rely on this variable when security is a concern.

BASH_ENV

Variable:	BASH_ENV	Since:	unknown
Type:	normal variable	Read-only:	no

Set by Bash:	no	Default:	n/a
---------------------	----	-----------------	-----

If this parameter is set when Bash is executing a shell script, its value is interpreted as a filename containing commands to initialize the shell, as in `~/ .bashrc` . The value of `BASH_ENV` is subjected to

- parameter expansion
- command substitution
- arithmetic expansion

before being interpreted as a file name.

`PATH` is not used to search for the resultant file name.

BASH_XTRACEFD

Variable:	BASH_XTRACEFD	Since:	4.1-alpha
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If set to an integer corresponding to a valid file descriptor, Bash will write the trace output generated when `set -x` is enabled to that file descriptor.

The file descriptor is closed when `BASH_XTRACEFD` is unset or assigned a new value.

Unsetting `BASH_XTRACEFD` or assigning it the empty string causes the trace output to be sent to the standard error. Note that setting `BASH_XTRACEFD` to 2 (the standard error file descriptor) and then unsetting it will result in the standard error being closed.

CDPATH

Variable:	CDPATH	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The search path for the `cd` builtin command.

This is a colon-separated list of directories in which the shell looks for destination directories specified by the `cd` command.

Example content:

```
.:~:/usr
```

COLUMNS

Variable:	COLUMNS	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	on SIGWINCH	Default:	n/a

Used by the select compound command to determine the terminal width when printing selection lists. Automatically set upon receipt of a SIGWINCH .

COMPREPLY

Variable:	COMPREPLY	Since:	unknown
Type:	integer indexed array	Read-only:	no
Set by Bash:	no	Default:	n/a

An array variable from which Bash reads the possible completions generated by a shell function invoked by the programmable completion facility.

EMACS

Variable:	EMACS	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If Bash finds this variable in the environment when the shell starts with value "t", it assumes that the shell is running in an Emacs shell buffer and disables line editing.

ENV

Variable:	ENV	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

Similar to BASH_ENV : Used when the shell is invoked in POSIX® mode.

FCEDIT

Variable:	FCEDIT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The default editor for the fc builtin command.

FIGNORE

Variable:	FIGNORE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

A colon-separated list of suffixes to ignore when performing filename completion. A filename whose suffix matches one of the entries in FIGNORE is excluded from the list of matched filenames.

Example content:

```
.o:~
```

FUNCNEST

Variable:	FUNCNEST	Since:	4.2-alpha
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If set to a numeric value greater than 0, defines a maximum function nesting level. Function invocations that exceed this nesting level will cause the current command to abort.

Negative values, 0 or non-numeric assignments have the effect as if FUNCNEST was unset or empty: No nest control

GLOBIGNORE

Variable:	GLOBIGNORE	Since:	2.0
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

A colon-separated list of patterns defining the set of filenames to be ignored by pathname expansion. If a filename matched by a pathname expansion pattern also matches one of the patterns in GLOBIGNORE , it is removed from the list of matches.

HISTCONTROL

Variable:	HISTCONTROL	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

A colon-separated list of values controlling how commands are saved on the history list:

ignorespace	lines which begin with a space character are not saved in the history list
ignoredups	don't save lines matching the previous history entry
ignoreboth	short for ignorespace:ignoredups
erasedups	remove all previous lines matching the current line from the history list before the current line is saved

Any value not in the above list is ignored.

If HISTCONTROL is unset, or does not include a valid value, all lines read by the shell parser are saved on the history list, subject to the value of HISTIGNORE . The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of HISTCONTROL .

HISTFILE

Variable:	HISTFILE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	~/.bash_history

The name of the file in which command history is saved.

If unset, the command history is not saved when an interactive shell exits.

HISTFILESIZE

Variable:	HISTFILESIZE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	HISTSIZE

The maximum number of lines contained in the history file.

When this variable is assigned a value, the history file is truncated, if necessary, by removing the oldest entries, to contain no more than the given number of lines. If the given number of lines is 0 (zero), the file is truncated to zero size. Non-numeric values and numeric values less than zero inhibit truncation.

The history file is also truncated to this size after writing it when an interactive shell exits.

HISTIGNORE

Variable:	HISTIGNORE	Since:	2.0
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

A colon-separated list of patterns used to decide which command lines should be saved on the history list. Each pattern is anchored at the beginning of the line and must match the complete line (no implicit '*' is appended).

Each pattern is tested against the line after the checks specified by HISTCONTROL are applied.

In addition to the normal shell pattern matching characters, "&" matches the previous history line. "&" may be escaped using a backslash; the backslash is removed before attempting a match.

The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of HISTIGNORE .

HISTSIZE

Variable:	HISTSIZE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	set at compile time (default 500)

The number of commands to remember in the command history.

If the number is set to 0 (zero), then the history list is disabled. If the number is set to any negative number, then the history list is unlimited.

HISTTIMEFORMAT

Variable:	HISTTIMEFORMAT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If this variable is set and not null, its value is used as a format string for strftime(3) to print the time stamp associated with each history entry displayed by the history builtin.

If this variable is set, time stamps are written to the history file so they may be preserved across shell sessions. This uses the history comment character to distinguish timestamps from other history lines.

HOME

Variable:	HOME	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The home directory of the current user.

The default argument for the `cd` builtin command.

The value of this variable is also used when performing tilde expansion.

HOSTFILE

Variable:	HOSTFILE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

Contains the name of a file in the same format as `/etc/hosts` that should be read when the shell needs to complete a hostname.

The list of possible hostname completions may be changed while the shell is running. the next time hostname completion is attempted after the value is changed, Bash adds the contents of the new file to the existing list.

If `HOSTFILE` is set, but has no value, or does not name a readable file, Bash attempts to read `/etc/hosts` to obtain the list of possible hostname completions.

When `HOSTFILE` is unset, the hostname list is cleared.

IFS

Variable:	IFS	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	<space><tab><newline>

The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the `read` builtin command.

IGNOREEOF

Variable:	IGNOREEOF	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	10 (when invalid)

Controls the action of an interactive shell on receipt of an `EOF(.)` character (e.g. by Ctrl-D) as the sole input.

If set, the value is the number of consecutive `EOF()` characters which must be typed as the first characters on an input line before Bash exits.

If the variable exists but does not have a numeric value, or has no value, the default value is 10.

If it does not exist, `EOF(.)` signifies the end of input to the shell.

INPUTRC

Variable:	INPUTRC	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The filename for the readline startup file, overriding the default of `~/.inputrc`.

LANG

Variable:	LANG	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

Used to determine the locale category for any category not specifically selected with a variable starting with `LC_`.

LC_ALL

Variable:	LC_ALL	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

This variable overrides the value of `LANG` and any other `LC_` variable specifying a locale category.

LC_COLLATE

Variable:	LC_COLLATE	Since:	unknown
Type:	normal variable	Read-only:	no

Set by Bash:	no	Default:	n/a
---------------------	----	-----------------	-----

This variable determines the collation order used when sorting the results of pathname expansion, and determines the behavior of range expressions, equivalence classes, and collating sequences within pathname expansion and pattern matching.

LC_CTYPE

Variable:	LC_CTYPE	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

This variable determines the interpretation of characters and the behavior of character classes within pathname expansion and pattern matching.

LC_MESSAGES

Variable:	LC_MESSAGES	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

This variable determines the locale used to translate double- quoted strings preceded by a \$.

LC_NUMERIC

Variable:	LC_NUMERIC	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

This variable determines the locale category used for number formatting.

LINES

Variable:	LINES	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	on SIGWINCH	Default:	n/a

Used by the select compound command to determine the column length for printing selection lists. Automatically set upon receipt of a SIGWINCH .

MAIL

Variable:	MAIL	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	system-dependent

If this parameter is set to a file or directory name and the MAILPATH variable is not set, Bash informs the user of the arrival of mail in the specified file or Maildir-format directory.

MAILCHECK

Variable:	MAILCHECK	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	60

Specifies how often (in seconds) Bash checks for mail.

When it is time to check for mail, the shell does so before displaying the primary prompt.

If this variable is unset, or set to a value that is not a number greater than or equal to zero, the shell disables mail checking.

MAILPATH

Variable:	MAILPATH	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	system-dependent

A colon-separated list of file names to be checked for mail.

The message to be printed when mail arrives in a particular file may be specified by separating the file name from the message with a '?' (question mark).

When used in the text of the message, \$_ expands to the name of the current mailfile.

Example content:

```
/var/mail/bfox?"You have mail":~/shell-mail?"$_ has mail!"
```

OPTERR

Variable:	OPTERR	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	yes	Default:	1 (set on startup)

If set to the value 1, Bash displays error messages generated by the `getopts` builtin command.

`OPTERR` is initialized to 1 each time the shell is invoked or a shell script is executed.

PATH

Variable:	PATH	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	system-dependent (set on compile time)

The search path for commands. This is a colon-separated list of directories in which the shell looks for commands.

A zero-length (null) directory name in the value of `PATH` indicates the current directory.

A null directory name may appear as two adjacent colons, or as an initial or trailing colon.

There can be a static path compiled in for use in a restricted shell.

POSIXLY_CORRECT

Variable:	POSIXLY_CORRECT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If this variable is in the environment when Bash starts, the shell enters posix mode before reading the startup files, as if the `-posix` invocation option had been supplied.

If it is set while the shell is running, Bash enables posix mode, as if the command `set -o posix` had been executed.

PROMPT_COMMAND

Variable:	PROMPT_COMMAND	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If set, the value is executed as a command prior to issuing each primary prompt.

PROMPT_COMMANDS

Variable:	PROMPT_COMMANDS	Since:	5.1-alpha
Type:	integer indexed array	Read-only:	no
Set by Bash:	no	Default:	n/a

If set, each element is executed as a command prior to issuing each primary prompt (like PROMPT_COMMAND , just as array).

PROMPT_DIRTRIM

Variable:	PROMPT_DIRTRIM	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If set to a number greater than zero, the value is used as the number of trailing directory components to retain when expanding the \w and \W prompt string escapes.

Characters removed are replaced with an ellipsis.

PS0

Variable:	PS0	Since:	4.4.0
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	" "

Expanded and displayed by interactive shells after reading a complete command but before executing it.

PS1

Variable:	PS1	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	" \s-\v\\$ "

The value of this parameter is expanded and used as the primary prompt string. See Controlling the Prompt (<https://www.gnu.org/software/bash/manual/bash.html#Controlling-the-Prompt>).

PS2

Variable:	PS2	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	" > "

The value of this parameter is expanded as with PS1 and used as the secondary prompt string.

PS3

Variable:	PS3	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The value of this parameter is used as the prompt for the select command.

PS4

Variable:	PS4	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	if unset	Default:	" + "

The value of this parameter is expanded as with PS1 and the value is printed before each command Bash displays during an execution trace. The first character of PS4 is replicated multiple times, as necessary, to indicate multiple levels of indirection.

SHELL

Variable:	SHELL	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The full pathname to the shell is kept in this environment variable. If it is not set when the shell starts, Bash assigns the full pathname of the current user's login shell.

SRANDOM

Variable:	SRANDOM	Since:	5.1-alpha
Type:	normal variable	Read-only:	no

Set by Bash:	yes	Default:	n/a
---------------------	-----	-----------------	-----

A variable that delivers a 32bit random number. The random number generation uses platform specific generators in the background and a builtin fallback generator.

TIMEFORMAT

Variable:	TIMEFORMAT	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The value of this parameter is used as a format string specifying how the timing information for pipelines prefixed with the time reserved word should be displayed.

The % character introduces an escape sequence that is expanded to a time value or other information. The escape sequences and their meanings are as follows, the braces denote optional portions:

%%	a literal % (percent sign)
%[p][l]R	elapsed time in seconds
%[p][l]U	number of CPU seconds spent in user mode
%[p][l]S	number of CPU seconds spent in system mode
%P	CPU percentage, computed as (%U + %S) / %R

The optional modifiers (p and l) are:

- p A digit specifying the precision. A value of 0 causes no decimal point or fraction to be output. At most three digits after the decimal point are shown. If not specified, the value 3 is used.
- l A longer format, including minutes, of the form MMmSS.FFs. The value of p determines whether or not the fraction is included.

If this variable is not set, Bash acts as if it had the value

```
$'\nreal\t%3lR\nuser\t%3lU\nsys%3lS'
```

If the value is null, no timing information is displayed.
A trailing newline is added when the format string is displayed.

TMOUT

Variable:	TMOUT	Since:	2.05b
Type:	normal variable	Read-only:	no

Set by Bash:	no	Default:	n/a
---------------------	----	-----------------	-----

If set to a value greater than zero, `TMOUT` is treated as the default timeout for the `read` builtin command.

The `select` command terminates if input does not arrive after `TMOUT` seconds when input is coming from a terminal.

In an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the primary prompt. Bash terminates after waiting for that number of seconds if input does not arrive.

TMPDIR

Variable:	TMPDIR	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

If set, Bash uses its value as the name of a directory in which Bash creates temporary files for the shell's use.

auto_resume

Variable:	auto_resume	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

This variable controls how the shell interacts with the user and job control. If this variable is set, single word simple commands without redirections are treated as candidates for resumption of an existing stopped job. There is no ambiguity allowed; if there is more than one job beginning with the string typed, the job most recently accessed is selected. The name of a stopped job, in this context, is the command line used to start it. If set to the value `exact`, the string supplied must match the name of a stopped job exactly; if set to `substring`, the string supplied needs to match a substring of the name of a stopped job. The `substring` value provides functionality analogous to the `%?` job identifier.

If set to any other value, the supplied string must be a prefix of a stopped job's name; this provides functionality analogous to the `%string` job identifier.

histchars

Variable:	histchars	Since:	unknown
Type:	normal variable	Read-only:	no
Set by Bash:	no	Default:	n/a

The two or three characters which control history expansion and tokenization.

The first character is the history expansion character, the character which signals the start of a history expansion, normally '!' (exclamation mark).

The second character is the quick substitution character, which is used as shorthand for re-running the previous command entered, substituting one string for another in the command. The default is '^' (caret).

The optional third character is the character which indicates that the remainder of the line is a comment when found as the first character of a word, normally '#' (hash mark). The history comment character causes history substitution to be skipped for the remaining words on the line. It does not necessarily cause the shell parser to treat the rest of the line as a comment.

Discussion

Chad Cloman (<http://www.cloman.com/>), [2011/04/24 04:18 \(\)](#)

You have a misspelling. It should be "loses" instead of "looses". The words have two distinct meanings.

Jan Schampera, [2011/04/24 14:00 \(\)](#)

Thank you, fixed. Indeed it changes the meaning in a weird way :)

Ilguz Latypov, [2012/01/27 06:14 \(\)](#)

I could find a way to set IFS temporarily around word splitting in an assignment.

```
$ ( IFS=":" m=$(echo "a:b"); echo ${m[1]}; echo -ne "${IFS}" | hexdump -C ) b
00000000 3a | 00000001
```

This limit becomes obvious after realizing that the assignment to m does not constitute a command, so Bash interprets IFS=... m=... as a series of assignments in the current shell level.

I wonder if Bash could introduce assignment of variables such as IFS around word splitting in arrays. It would look like this,



```
m=([IFS=":"]echo "a:b")
```

So far, I worked around the limit by using the read built-in,

```
$ ( IFS=":" read -r m1 m2 << "a:b"; echo "${m2}"; echo -ne "${IFS}" | hexdump -C ) b
00000000 20 09 0a | .. 00000003
```

swaroop vasireddy, [2015/09/16 11:20 \(\)](#)

Is there a way that getopts will be able to read contents of a file (i.e) declared variables in bash script ? If so, could you let me know with an example. This would be a great help for me. Very nice article and appreciate your effort here.

 syntax/shellvars.txt  Last modified: 2023/01/20 05:39 by fgrose

This site is supported by Performing Databases - your experts for database administration

Bash Hackers Wiki



Except where otherwise noted, content on this wiki is licensed under the following license:
GNU Free Documentation License 1.3