NAME

set, unset, setenv, unsetenv, export – shell built-in functions to determine the characteristics for environmental variables of the current shell and its descendents

SYNOPSIS

sh

set [**––aefhkntuvx** [*argument* . . .]]

unset [*name* . . .]

export [*name* . . .]

csh

set [*var* [**=** *value*]]
set *var*[*n*] **=** *word*

unset *pattern*

setenv [*VAR* [*word*]]

unsetenv *variable*

ksh

set [±**aefhkmnopstuvx**] [±**o** *option*]. . . [±**A** *name*] [*arg* . . .]

unset [–**f**] *name* . . .

†† **export** [*name*[=*value*]] . . .

DESCRIPTION

sh

The **set** built-in command has the following options:

–a Mark variables which are modified or created for export.

–e Exit immediately if a command exits with a non-zero exit status.

–f Disable file name generation.

–h Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).

–k All keyword arguments are placed in the environment for a command, not just those that precede the command name.

–n Read commands but do not execute them.

–t Exit after reading and executing one command.

–u Treat unset variables as an error when substituting.

–v Print shell input lines as they are read.

–x Print commands and their arguments as they are executed.

– – Do not change any of the flags; useful in setting **$1** to –.

Using **+** rather than – causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **$–**.
The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, **. . . .** If no arguments are given the values of all names are printed.

For each *name*, **unset** removes the corresponding variable or function value. The variables **PATH, PS1, PS2, MAILCHECK,** and **IFS** cannot be unset.

With the **export** built-in, the given *name*s are marked for automatic export to the *environment* of subsequently executed commands. If no arguments are given, variable names that have been marked for export during the current shell's execution are listed. (Variable names exported from a parent shell are listed only if they have been exported again during the current shell's execution.) Function names are *not* exported.

csh

With no arguments, **set** displays the values of all shell variables. Multiword values are displayed as a parenthesized list. With the *var* argument alone, **set** assigns an empty (null) value to the variable *var*. With arguments of the form *var* **=** *value* **set** assigns *value* to *var*, where *value* is one of:

word	A single word (or quoted string).
(*wordlist***)**	A space-separated list of words enclosed in parentheses.

Values are command and filename expanded before being assigned. The form **set** *var***[***n***]** **=** *word* replaces the *n*'th word in a multiword value with *word*.

unset removes variables whose names match (filename substitution) *pattern*. All variables are removed by '**unset** *'; this has noticeably distasteful side effects.

With no arguments, **setenv** displays all environment variables. With the *VAR* argument, **setenv** sets the environment variable *VAR* to have an empty (null) value. (By convention, environment variables are normally given upper-case names.) With both *VAR* and *word* arguments, **setenv** sets the environment variable **NAME** to the value *word*, which must be either a single word or a quoted string. The most commonly used environment variables, USER, TERM, and PATH, are automatically imported to and exported from the **csh** variables **user**, **term**, and **path**; there is no need to use **setenv** for these. In addition, the shell sets the **PWD** environment variable from the **csh** variable **cwd** whenever the latter changes.

The environment variables **LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY** take immediate effect when changed within the C shell.

If any of the **LC_*** variables (**LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **csh** for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_*** variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how **csh** behaves.

LC_CTYPE

Determines how **csh** handles characters. When **LC_CTYPE** is set to a valid value, **csh** can display and handle text and filenames containing valid characters for that locale. **csh** can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **csh** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES
Determines how diagnostic and informative messages are presented. This includes the language and style of the messages and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S./English).

LC_NUMERIC
Determines the value of the radix character (decimal point (".") in the "C" locale) and thousand separator (empty string ("") in the "C" locale).

unsetenv removes *variable* from the environment. As with **unset**, pattern matching is not performed.

ksh

The flags for the **set** built-in have meaning as follows:

–A Array assignment. Unset the variable *name* and assign values sequentially from the list *arg*. If **+A** is used, the variable *name* is not unset first.

–a All subsequent variables that are defined are automatically exported.

–e If a command has a non-zero exit status, execute the **ERR** trap, if set, and exit. This mode is disabled while reading profiles.

–f Disables file name generation.

–h Each command becomes a tracked alias when first encountered.

–k All variable assignment arguments are placed in the environment for a command, not just those that precede the command name.

–m Background jobs will run in a separate process group and a line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.

–n Read commands and check them for syntax errors, but do not execute them. Ignored for interactive shells.

–o The following argument can be one of the following option names:

allexport
Same as **–a**.

errexit Same as **–e**.

bgnice All background jobs are run at a lower priority. This is the default mode. **emacs** Puts you in an **emacs** style in-line editor for command entry.

gmacs Puts you in a **gmacs** style in-line editor for command entry.

ignoreeof
The shell will not exit on end-of-file. The command **exit** must be used.

keyword
Same as **–k**.

markdirs

All directory names resulting from file name generation have a trailing **/** appended.

monitor Same as −**m**.

noclobber
Prevents redirection **>** from truncating existing files. Require **>|** to truncate a file when turned on.

noexec Same as −**n**.

noglob Same as −**f**.

nolog Do not save function definitions in history file.

nounset Same as −**u**.

privileged
Same as −**p**.

verbose Same as −**v**.

trackall Same as −**h**.

vi Puts you in insert mode of a **vi** style in-line editor until you hit escape character **033**. This puts you in control mode. A return sends the line.

viraw Each character is processed as it is typed in **vi** mode.

xtrace Same as −**x**.

If no option name is supplied then the current option settings are printed.

−**p** Disables processing of the **$HOME/.profile** file and uses the file **/etc/suid_profile** instead of the **ENV** file. This mode is on whenever the effective uid is not equal to the real uid, or when the effective gid is not equal to the real gid. Turning this off causes the effective uid and gid to be set to the real uid and gid.

−**s** Sort the positional parameters lexicographically.

−**t** Exit after reading and executing one command.

−**u** Treat unset parameters as an error when substituting.

−**v** Print shell input lines as they are read.

−**x** Print commands and their arguments as they are executed.

− Turns off −**x** and −**v** flags and stops examining arguments for flags.

− − Do not change any of the flags; useful in setting **$1** to a value beginning with −. If no arguments follow this flag then the positional parameters are unset.

Using **+** rather than − causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **$**−. Unless −**A** is specified, the remaining arguments are positional parameters and are assigned, in order, to **$1 $2** . . .. If no arguments are given then the names and values of all variables are printed on the standard output.

The variables given by the list of *name*s are unassigned, i.e., their values and attributes are erased. **readonly** variables cannot be unset. If the −**f**, flag is set, then the names refer to *function* names. Unsetting **ERRNO**, **LINENO**, **MAILCHECK**, **OPTARG**, **OPTIND**, **RANDOM**,

SECONDS, **TMOUT**, and _ removes their special meaning even if they are subsequently assigned.

When using **unset**, the variables given by the list of *name*s are unassigned, i.e., their values and attributes are erased. **readonly** variables cannot be unset. If the **−f**, flag is set, then the names refer to *function* names. Unsetting **ERRNO**, **LINENO**, **MAILCHECK**, **OPTARG**, **OPTIND**, **RANDOM**, **SECONDS**, **TMOUT**, and _ removes their special meaning even if they are subsequently assigned.

With the **export** built-in, the given *name*s are marked for automatic export to the **environment** of subsequently-executed commands.

On this man page, **ksh**(1) commands that are preceded by one or two † (daggers) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by †† that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the **=** sign and word splitting and file name generation are not performed.

SEE ALSO

csh(1), **ksh**(1), **read**(1), **sh**(1), **typeset**(1)