The set builtin command

Fix Me! incomplete - text, examples, maybe extended description

Synopsis

```
set [--abefhkmnptuvxBCHP] <-o OPTIONNAME> [-][--] <POSPARAMS>
```

Description

`set` is primarily made to

- set the positional parameters (see handling positional parameters) to `<POSPARAMS>`
- set shell attributes with short options (see below)
- set shell attributes with long option names (see below)

Without any options, `set` displays all shell- and environment-variables (only is POSIX-mode) in a re-usable format `NAME=VALUE`.

Attributes

All attributes below can be switched on using `-X` and switched off using `+X`. This is done because of the historical meaning of the `-` to set flags (true for most commands on UNIX®).

Flag	Optionname	Description
`-a`	`allexport`	Automatically mark new and altered variables to be exported to subsequent environments.
`-b`	`notify`	Don't wait for the next prompt to print when showing the reports for a terminated background job (only with job control)
`-e`	`errexit`	When set, the shell exits when a simple command in a command list exits non-zero (`FALSE`). This is not done in situations, where the exit code is already checked (`if`, `while`, `until`, `||`, `&&`)
`-f`	`noglob`	Disable pathname expansion (globbing)
`-h`	`hashall`	Remembers the location of commands when they're called (hashing). Enabled by default.
`-k`	`keyword`	Allows to place environment-assignments everywhere in the commandline, not only infront of the called command.
`-m`	`monitor`	**Monitor mode**. With job control, a short descriptive line is printed when a backgroud job ends. Default is "on" for interactive shells (with job control).
`-n`	`noexec`	Read and parse but **do not execute commands** - useful for checking scripts for syntax errors. Ignored by interactive shells.
`-o`		Set/unset attributes with long option names, e.g. `set -o noglob`. The long option names are in the second column of this table. If no option name is given, all options are printed with their current status.
`-p`	`privileged`	Turn on privileged mode.
`-t`	`onecmd`	Exit after reading and executing **one** command.
`-u`	`nounset`	Treat unset variables as an error when performing parameter expansion. Non-interactive shells exit on this error.
`-v`	`verbose`	Print shell input lines as they are read - useful for debugging.
`-x`	`xtrace`	Print commands just before execution - with all expansions and substitutions done, and words marked - useful for debugging.
		The shell performs brace expansion This is on by default.

-B	braceexpand	
-C	noclobber	Don't overwrite files on redirection operations. You can override that by specifying the `>|` redirection operator when needed. See redirection
-E	errtrace	ERR-traps are inherited by by shell functions, command substitutions, and commands executed in a subshell environment.
-H	histexpand	Enable `!`-style history expansion. Defaults to `on` for interactive shells.
-P	physical	Don't follow symlinks when changing directories - use the physical filesystem structure.
-T	functrace	DEBUG- and RETURN-traps are inherited by subsequent environments, like `-E` for ERR trap.
-		"End of options" - all following arguments are assigned to the positional parameters, even when they begin with a dash. `-x` and `-v` options are turned off. Positional parameters are unchanged (unlike using `--`!) when no further arguments are given.
--		If no arguments follow, the positional parameters are unset. With arguments, the positional parameters are set, even if the strings begin with a `-` (dash) like an option.
Long options usable with `-o` without a short equivalent		
	emacs	Use an emacs-style command line editing interface. This is enabled by default when the shell is interactive, unless the shell is started with `–noediting` option.
	history	If set, command historization is done (enabled by default on interactive shells)
	ignoreeof	The effect is as if the shell command `IGNOREEOF=10` had been executed. See shell variables.
	nolog	**(currently ignored)**
	pipefail	If set, the exit code from a pipeline is different from the normal ("last command in pipeline") behaviour: `TRUE` when no command failed, `FALSE` when something failed (code of the rightmost command that failed)
	posix	When set, Bash runs in POSIX mode.
	vi	Enables a `vi`-style command line editing interface.

Examples

Tag a part of a shell script to output debugging information (`-x`):

```
#!/bin/bash
...
set -x # on
...
set +x # off
...
```

Portability considerations

`set` and its basic behaviour and options are specified by POSIX®. However, options that influence Bash-specific things are not portable, naturally.

See also

- Internal: The shopt builtin command

Discussion

commands/builtin/set.txt · Last modified: 2011/03/21 03:50 by fgrose

Fatal error: Out of memory (allocated 33816576) (tried to allocate 40 bytes) in **/homepages/41/d13525936/htdocs/bash-hackers-**