Bash 4 - a rough overview

Attention: Since Bash 4 has been around for quite some time now (4.3 will come soon), I consider it to be "standard". This page is not maintained anymore and is left here to keep your links working. See the Bash changes page for new stuff introduced.
Besides many bugfixes since Bash 3.2, Bash 4 will bring some interesting new features for shell users and scripters. See also Bash changes for a small general overview with more details.

Not all of the changes and news are included here, just the biggest or most interesting ones. The changes to completion, and the readline component are not covered. **Though, if you're familiar with these parts of Bash (and Bash 4), feel free to write a chapter here.**

The complete list of fixes and changes is in the CHANGES or NEWS file of your Bash 4 distribution.

The current available **stable** version is 4.4.18 release (February 03, 2018):

- ftp://ftp.cwru.edu/pub/bash/bash-4.4.18.tar.gz (ftp://ftp.cwru.edu/pub/bash/bash-4.4.18.tar.gz)
- ftp://ftp.gnu.org/pub/gnu/bash/bash-4.4.18.tar.gz (ftp://ftp.gnu.org/pub/gnu/bash/bash-4.4.18.tar.gz)

New or changed commands and keywords

The new "coproc" keyword

Bash 4 introduces the concepts of coprocesses, a well known feature of other shells. The basic concept is simple: It will start any command in the background and set up an array that is populated with accessible files that represent the filedescriptors of the started process.

In other words: It lets you start a process in background and communicate with its input and output data streams.

See The coproc keyword

The new "mapfile" builtin

The `mapfile` builtin is able to map the lines of a file directly into an array. This avoids having to fill an array yourself using a loop. It enables you to define the range of lines to read, and optionally call a callback, for example to display a progress bar.

See: The mapfile builtin command

Changes to the "case" keyword

The `case` construct understands two new action list terminators:

The `;&` terminator causes execution to continue with the next action list (rather than terminate the `case` construct).

The `;;&` terminator causes the `case` construct to test the next given pattern instead of terminating the whole execution.

See The case statement

Changes to the "declare" builtin

The `-p` option now prints all attributes and values of declared variables (or functions, when used with `-f`). The output is fully re-usable as input.

The new option `-l` declares a variable in a way that the content is converted to lowercase on assignment. For uppercase, the same applies to `-u`. The option `-c` causes the content to be capitalized before assignment.

`declare -A` declares associative arrays (see below).

Changes to the "read" builtin

The `read` builtin command has some interesting new features.

The `-t` option to specify a timeout value has been slightly tuned. It now accepts fractional values and the special value 0 (zero). When `-t 0` is specified, `read` immediately returns with an exit status indicating if there's data waiting or not. However, when a timeout is given, and the `read` builtin times out, any partial data recieved up to the timeout is stored in the given variable, rather than lost. When a timeout is hit, `read` exits with a code greater than 128.

A new option, `-i`, was introduced to be able to preload the input buffer with some text (when Readline is used, with `-e`). The user is able to change the text, or press return to accept it.

See The read builtin command

Changes to the "help" builtin

The builtin itself didn't change much, but the data displayed is more structured now. The help texts are in a better format, much easier to read.

There are two new options: `-d` displays the summary of a help text, `-m` displays a manpage-like format.

Changes to the "ulimit" builtin

Besides the use of the 512 bytes blocksize everywhere in POSIX mode, `ulimit` supports two new limits: `-b` for max socket buffer size and `-T` for max number of threads.

Expansions

Brace Expansion

The brace expansion was tuned to provide expansion results with leading zeros when requesting a row of numbers.

See Brace expansion

Parameter Expansion

Methods to modify the case on expansion time have been added.

On expansion time you can modify the syntax by adding operators to the parameter name.

See Case modification on parameter expansion

Substring expansion

When using substring expansion on the positional parameters, a starting index of 0 now causes $0 to be prepended to the list (if the positional parameters are used). Before, this expansion started with $1:

```
# this should display $0 on Bash v4, $1 on Bash v3
echo ${@:0:1}
```

Globbing

There's a new shell option `globstar`. When enabled, Bash will perform recursive globbing on `**` – this means it matches all directories and files from the current position in the filesystem, rather than only the current level.

The new shell option `dirspell` enables spelling corrections on directory names during globbing.

See Pathname expansion (globbing)

Associative Arrays

Besides the classic method of integer indexed arrays, Bash 4 supports associative arrays.

An associative array is an array indexed by an arbitrary string, something like

```
declare -A ASSOC

ASSOC[First]="first element"
ASSOC[Hello]="second element"
ASSOC[Peter Pan]="A weird guy"
```

See Arrays

Redirection

There is a new `&>>` redirection operator, which appends the standard output and standard error to the named file. This is the same as the good old `>>FILE 2>&1` notation.

The parser now understands `|&` as a synonym for `2>&1 |`, which redirects the standard error for a command through a pipe.

See Redirection

Interesting new shell variables

Variable	Description
BASHPID	contains the PID of the current shell (this is different than what `$$` does!)
PROMPT_DIRTRIM	specifies the max. level of unshortened pathname elements in the prompt
FUNCNEST	control the maximum number of shell function recursions

See Special parameters and shell variables

Interesting new Shell Options

The mentioned shell options are **off by default** unless otherwise mentioned.

Option	Description
`checkjobs`	check for and report any running jobs at shell exit
`compat*`	set compatiblity modes for older shell versions (influences regular expression matching in `[[ ... ]]`
`dirspell`	enables spelling corrections on directory names during globbing
`globstar`	enables recursive globbing with `**`
`lastpipe`	(4.2) to execute the last command in a pipeline in the current environment

See List of shell options

Misc

- If a command is not found, the shell attempts to execute a shell function named `command_not_found_handle`, supplying the command words as the function arguments. This can be used to display userfriendly messages or perform different command searches
- The behaviour of the `set -e` (`errexit`) mode was changed, it now acts more intuitive (and is better documented in the manpage).
- The output target for the `xtrace` (`set -x` / `set +x`) feature is configurable **since Bash 4.1** (previously, it was fixed to `stderr`): a variable named BASH_XTRACEFD can be set to the filedescriptor that should get the output
- Bash 4.1 is able to log the history to syslog (only to be enabled at compile time in `config-top.h`)

Discussion

bash4.txt Last modified: 2018/02/03 07:52 by narutowindy