What is the essential difference between compound command and normal command in bash?

0

I am learning bash. Now I learning [[...]] command and ((...)) command. They are called compound command, which is discriminated from normal command, like [. I read the article of "Compound Commands" in bash manual. It seems they runs with their own rules. But unfortunately, I could not think of why they are called as "compound". Someone who knows it, please let me know. Thank you very much.

bash

asked Aug 29 '16 at 5:52

mora

941 • 7 • 21

1 See: Difference in Bash between IF statements with parenthesis and square brackets – Cyrus Aug 29 '16 at 5:54

1 You seem to be mixing up terminology big time here. `((...))` is arithmetic evaluation, not compound commands. The difference between `[` and `[[` is that the old POSIX `[` is specified elsewhere, and so where Bash wants functionality which differs from POSIX, it can't use that; so there is a separate `[[` built-in which basically overshadows the old `[`. – tripleee Aug 29 '16 at 6:06

1 Voting to close as unclear what you are asking. – tripleee Aug 29 '16 at 6:06

4 @tripleee: `man bash` lists them under "Compound Commands" as they have different parsing rules. – choroba Aug 29 '16 at 6:10

1 @tripleee : Thank you for comments. And I am sorry for my unclear question. I found ((expression)) and [[expression]] under "Compound Commands" in bash manual. I need to tell you that I know the functional difference between [and [[. My question is what "compound" means. I can find a word, "compound", in dictionaries but I cannot think of what it actually means in "compound" command. I am just curious why they are called so. Thank you very much. – mora Aug 29 '16 at 6:22

1 Answer

5

Bash distinguishes between simple commands and compound commands:

1. **Simple commands** are a single command with optional arguments and redirections. For example:

```
$ echo Hello
Hello
```

2. **Compound commands** combine one or more simple commands into something that functions as a single unit. For example:

```
$ { echo Hello; date; }
Hello
Sun Aug 28 23:16:03 PDT 2016
```

One useful feature of compound commands is that redirections applied to the compound command are applied to every command that it contains. For example:

```
$ { echo info1; echo info2; } >logfile
$ cat logfile
info1
info2
```

According to `man bash`, there are four types of compound commands:

1. **Group:** `{...;}`, as illustrated above can be used to group simple commands together to form a compound command.
2. **Subshell:** `(...)` is similar to a group except that the commands are run in subshell environment. This means that variable assignments do not survive after the subshell completes. As an example:

```
$ a=0; (a=10; echo "inside=$a"); echo "outside=$a"
inside=10
outside=0
```

3. **Arithmetic Expression:** Inside double-parens, a series of comma-separated arithmetic calculations may be performed. For example:

```
$ ((a=2, a=10*a, a+=2)); echo "a=$a"
a=22
```

4. **Test Command:** Bash's advanced form of the test command, `[[...]]`, can include several tests. Tests are separated by `&&` or `||`:

```
$ [[ a == b || 3 -gt 2 && 4 -gt 3 ]]; echo $?
0
```

Documentation

From `man bash`:

> *Compound Commands*
>
> A compound command is one of the following. In most cases a list in a command's description may be separated from the rest of the command by one or more newlines, and may be followed by a newline in place of a semicolon.
>
> `(list)`
>
> list is executed in a subshell environment (see COMMAND EXECUTION ENVIRONMENT below). Variable assignments and builtin commands that affect the shell's environment do not remain in effect after the command completes. The return status is the exit status of list.
>
> `{ list; }`
>
> list is simply executed in the current shell environment. list must be terminated with a newline or semicolon. This is known as a group command. The return status is the exit status of list. Note that unlike the metacharacters (and), { and } are reserved words and must occur where a reserved word is permitted to be recognized. Since they do not cause a word break, they must be separated from list by whitespace or
> another shell metacharacter.
>
> `((expression))`
>
> The expression is evaluated according to the rules described below under ARITHMETIC EVALUATION. If the value of the expression is non-zero, the return status is 0; otherwise the return status is 1. This is exactly equivalent to let "expression".
>
> `[[ expression ]]`
>
> Return a status of 0 or 1 depending on the evaluation of the conditional expression expression. Expressions are composed of the primaries described below under CONDITIONAL EXPRESSIONS. Word splitting and pathname expansion are not performed on the words between the [[and]]; tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process

substitution, and quote removal are performed. Conditional operators such as -f must be unquoted to be recognized as primaries.

When used with [[, the < and > operators sort lexicographically using the current locale.

edited Aug 29 '16 at 6:53

answered Aug 29 '16 at 6:40

John1024

81.9k ● 9 ● 75 ● 104

2 Thank you for detailed and clear instruction. I could not contrast simple command with compound command before your answer. That is the reason I was confused. Thank you again. – mora Aug 29 '16 at 6:58