

Why am I getting a 'unary operator expected' error?

Asked 11 years, 1 month ago Modified 1 year, 7 months ago Viewed 97k times



25



I'm writing a shell script to streamline my development workflow.

It takes an argument as to which theme folder I'm going to be working in and starts `grunt watch` on that directory.

If I call the script without the necessary argument I'm currently printing a warning that a theme needs to be specified as a command line argument.

I'd like to print a list of the available options, e.g. theme directories

This is what I have so far...

```
THEME=$1

if [ $THEME == '' ]
then
    echo 'Need to specify theme'
else
    cd 'workspace/aws/ghost/'$THEME/'
    grunt watch
fi
```

Ideally I'd replace the output of the `echo` line with an `ls` of the themes parent directory like so

```
THEME=$1

if [ $THEME == '' ]
then
    echo 'Need to specify theme from the following'
    ls workspace/aws/ghost
else
    cd 'workspace/aws/ghost/'$THEME/'
    grunt watch
fi
```

However this gives me the following error

```
./ghost_dev.sh: line 3: [: ==: unary operator expected
```

bash

shell

Share

edited Oct 30, 2013 at 18:48

asked Oct 30, 2013 at 18:32

Improve this question

Follow



Luke

3,541 ● 6 ● 40 ● 63

2 You should use `-z` to check for an empty variable: `if [-z "$THEME"]`. – Kevin Oct 30, 2013 at 18:48

edited the title, I'll try with the `-z` – Luke Oct 30, 2013 at 18:48

1 Tip: [ShellCheck](#) would automatically tell you that you need double quotes around `$THEME`. – that other guy Oct 30, 2013 at 20:19

3 Answers

Sorted by: Highest score (default)



You need quotes around `$THEME` here:

58

```
if [ $THEME == '' ]
```



Otherwise, when you don't specify a theme, `$THEME` expands to nothing, and the shell sees this syntax error:



```
if [ == '' ]
```



With quotes added, like so:

```
if [ "$THEME" == '' ]
```

the expansion of an empty `$THEME` yields this valid comparison instead:

```
if [ "" == '' ]
```

This capacity for runtime syntax errors can be surprising to those whose background is in more traditional programming languages, but command shells (at least those in the Bourne tradition) parse code somewhat differently. In many contexts, shell parameters behave more like macros than variables; this behavior provides flexibility, but also creates traps for the unwary.

Since you tagged this question **bash**, it's worth noting that there is no word-splitting performed on the result of parameter expansion inside the "new" test syntax available in bash (and ksh/zsh), namely `[[...]]`. So you can also do this:

```
if [[ $THEME == '' ]]
```

The places you can get away without quotes are listed [here](#). But it's a fine habit to always quote parameter expansions anyway except when you explicitly want word-splitting (and even then, look to see if arrays will solve your problem instead).

It would be more idiomatic to use the `-z` test operator instead of equality with the empty string:

```
if [ -z "$THEME" ]
```

Of course, `[[...]]` doesn't require any quotes for this version, either:

```
if [[ -z $THEME ]]
```

But `[[...]]` is not part of the POSIX standard; for that matter, neither is `==`. So if you care about strict compatibility with other POSIX shells, stick to the quoting solution and use either `-z` or a single `=`.

Share

edited May 22, 2023 at 19:58

answered Oct 30, 2013 at 18:33

Improve this answer



Mark Reed

95k ● 17 ● 141 ● 185

Follow

- 1 OK, I've added the quotes around the variable assignment, used the `[[]]` syntax and added the quotes on `"$THEME"` for best practices. Works a treat! Thanks. – [Luke](#) Oct 30, 2013 at 18:46
- 3 Assignments work a little differently; at least word splitting does not occur on the RHS, so I think `THEME="$1"` and `THEME=$1` are identical, although I'd love to be corrected for the corner case I am forgetting. – [chepner](#) Oct 30, 2013 at 20:48

@chepner - you're right, no need for quotes around expansions on the right-hand side of an assignment. Still need to quote any literal whitespace, of course. – [Mark Reed](#) Mar 11, 2016 at 3:20

4 `["$THEME"]` will evaluate to false if `$THEME` is undefined or an empty string and true otherwise. See http://www.gnu.org/software/bash/manual/html_node/Bash-Conditional-Expressions.html#Bash-Conditional-Expressions. You can rearrange your if statement to exploit this behavior and have an even simpler conditional:

```
if [ "$THEME" ]; then
    cd 'workspace/aws/ghost/'$THEME/'
    grunt watch
else
```

```
echo 'Need to specify theme from the following'
ls workspace/aws/ghost
fi
```

`"$THEME"` needs to be in double-quotes, in case its value contains whitespace.

Share Improve this answer Follow

answered Oct 30, 2013 at 20:04



Digital Trauma

16k ● 4 ● 53 ● 85



Please correct the syntax with the double quotes.

-1



```
if [ "$THEME" == "" ]; then
    echo 'Need to specify theme from the following'
    ls workspace/aws/ghost
fi
```



Share

Improve this answer

Follow

edited Feb 15, 2019 at 15:37



Mark J. Bobak

14.3k ● 6 ● 43 ● 72

answered Oct 30, 2013 at 18:41



Prem Sriram

190 ● 5

Please correct the syntax of your if-statement. There are two errors. When 'then' is in the same line like 'if', you have to use a semicolon. Or just write the 'then' in the next line. And the enclosing 'if' is also missing. Please test your answer before posting it. Just imagine when a beginner is trying your answer. He will get a couple of error messages. – Sedat Kilinc Jun 27, 2018 at 20:19

I cleaned up the answer. – Mark J. Bobak Feb 15, 2019 at 15:38