Difference between [and [[in Shell Scripts

Finnbarr P. Murphy

(fpm@fpmurphy.com)

Many shell script writers are aware that *[* is a synonym for the *test* command (See the *test(1)* man page) but few are fully aware of the actual differences between *[* and *[[*.

The *test* command implements the portable syntax specified by the IEEE Std 1003.1-2008 (POSIX) standard. In most shells (older Bourne shells being the main exception), *[* is a synonym for *test* (but requires a final argument of]) and is a shell builtin. In addition, there is usually an external executable of that name, e.g. */bin/[* or */usr/bin/[*.

The IEEE 1003.1-2008 standard (POSIX) defines a mandatory feature set for the *test* utility but almost every shell implements extensions to the standard. If you want fully portable shell scripts, you should be careful not to use any non-POSIX extensions.

Unlike the simple *test* command, *[[ ]]* is a conditional shell expression. Double brackets implicitly quote all arguments, so you do not need to add quotes yourself. No word splitting or glob expansion is done by default. The double bracket conditional expression also uses a more C-like syntax, supporting all the usual operators as well as && instead of -a (logical AND), and || instead of -o (logical OR). It can also do glob-like string matching using the ~= syntax. Note that parentheses also do not need to be escaped.

Compare the following expression:

```
[ -f "$file" -a ( -d "$dir1" -o -d "$dir2" ) ]
```

with:

```
[[ -f $file && ( -d $dir1 || -d $dir2) ]]
```

The double bracket conditional expression is simpler to write and easier to understand.

Another advantage of the double bracket conditional expression is that in *ksh93* and the latest versions of the *bash* shell, string comparisons using (greater than) respect the current locale collating order (*LC_COLLATE*.)

There is one case where quoting will still make a difference inside a double bracket conditional expression and that is in regards to pattern matching. For example:

```
[[ $str == foo* ]]
```

is true if the *str* variable starts with *foo* but in the following conditional expression

```
[[ $str == "foo*" ]]
```


is only true if the *str* variable contains the literal four character string *foo**.

