I'm confused with using single or double bracket [[]]. look at code:

dir="/home/mazimi/VirtualBox VMs"

if [[ -d ${dir} ]]; then

echo "yep"

fi

It works perfectly although the string contains space. But when I change it to single bracket []:

dir="/home/mazimi/VirtualBox VMs"

if [ -d ${dir} ]; then

echo "yep"

fi

It says:

./script.sh: line 5: [: /home/mazimi/VirtualBox: binary operator expected

When I change it to:

dir="/home/mazimi/VirtualBox VMs"

if [ -d "${dir}" ]; then

echo "yep"

fi

It works fine. Can someone explain what is happening? When should I assign double quotes around variables like "${var}" to prevent problems caused by spaces?

The single bracket [ is actually an alias for the test command, it's not syntax.

One of the downsides (of many) of the single bracket is that if one or more of the operands it is trying to evaluate return an empty string, it will complain that it was expecting two operands (binary). This is why you see people do [ x$foo = x$blah ], the x guarantees that the operand will never evaluate to an empty string.

The double bracket [[ ]], on the other hand, is syntax and is much more capable than [ ]. As you found out, it does not have the single operand issue and it also allows for more C-like syntax with >, <, >=, <=, !=, ==, &&, || operators.

My recommendation is the following: If your interpreter is #!/bin/bash, then always use [[ ]]

It is important to note that [[ ]] is not supported by all POSIX shells, however many shells do support it such as zsh and ksh in addition to bash