An if statement typically looks like

if commands1

then

commands2

else

commands3

fi

The then clause is executed if the exit code of commands1 is zero. If the exit code is nonzero, then the else clause is executed. commands1 can be simple or complex. It can, for example, be a sequence of one or more pipelines separated by one of the operators ;, &, &&, or ||. The if conditions shown below are just special cases of commands1:

1. if [ condition ]

This is the traditional shell test command. It is available on all POSIX shells. The test command sets an exit code and the if statement acts accordingly. Typical tests are whether a file exists or one number is equal to another.

1. if [[ condition ]]

This is a new upgraded variation on test from *ksh* that *bash* and *zsh* also support. This test command also sets an exit code and the if statement acts accordingly. Among its extended features, it can test whether a string matches a regular expression.

1. if ((condition))

Another *ksh* extension that *bash* and *zsh* also support. This performs arithmetic. As the result of the arithmetic, an exit code is set and the if statement acts accordingly. It returns an exit code of zero (true) if the result of the arithmetic calculation is nonzero. Like [[...]], this form is not POSIX and therefore not portable.

1. if (command)

This runs command in a subshell. When command completes, it sets an exit code and the if statement acts accordingly.

A typical reason for using a subshell like this is to limit side-effects of command if command required variable assignments or other changes to the shell's environment. Such changes do not remain after the subshell completes.

1. if command

command is executed and the if statement acts according to its exit code.